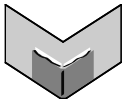
The Journal of Machine Learning Research
Volume 10
Print-Archive Edition

Pages 1485–2962



Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 10
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 2009.

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 Editorial Board

Journal of Machine Learning Research

Volume 10, 2009

Journal of Machine Learning Research 10 (2009) 1485-1510 Submitted 3/08; Revised 10/08; Published 7/09

Robustness and Regularization of Support Vector Machines

Huan Xu XUHUAN@CIM.MCGILL.CA
Department of Electrical and Computer Engineering
3480 University Street
McGill University
Montreal, Canada H3A 2A7

Constantine Caramanis CARAMANIS@MAIL.UTEXAS.EDU
Department of Electrical and Computer Engineering
The University of Texas at Austin
1 University Station C0803
Austin, TX, 78712, USA

Shie Mannor* SHIE.MANNOR@MCGILL.CA
Department of Electrical and Computer Engineering
3480 University Street
McGill University
Montreal, Canada H3A 2A7

Editor: Alexander Smola

Abstract

We consider regularized support vector machines (SVMs) and show that they are precisely equivalent to a new robust optimization formulation. We show that this equivalence of robust optimization and regularization has implications for both algorithms, and analysis. In terms of algorithms, the equivalence suggests more general SVM-like algorithms for classification that explicitly build in protection to noise, and at the same time control overfitting. On the analysis front, the equivalence of robustness and regularization provides a robust optimization interpretation for the success of regularized SVMs. We use this new robustness interpretation of SVMs to give a new proof of consistency of (kernelized) SVMs, thus establishing robustness as the *reason* regularized SVMs generalize well.

Keywords: robustness, regularization, generalization, kernel, support vector machine

1. Introduction

Support Vector Machines (SVMs for short) originated in Boser et al. (1992) and can be traced back to as early as Vapnik and Lerner (1963) and Vapnik and Chervonenkis (1974). They continue to be one of the most successful algorithms for classification. SVMs address the classification problem by finding the hyperplane in the feature space that achieves maximum sample margin when the training samples are separable, which leads to minimizing the norm of the classifier. When the samples are not separable, a penalty term that approximates the total training error is considered (Bennett and Mangasarian, 1992; Cortes and Vapnik, 1995). It is well known that minimizing the training error itself can lead to poor classification performance for new unlabeled data; that is, such an approach

∗. Also at the Department of Electrical Engineering, Technion, Israel.

may have poor generalization error because of, essentially, overfitting (Vapnik and Chervonenkis, 1991). A variety of modifications have been proposed to handle this, one of the most popular methods being that of minimizing a combination of the training-error and a regularization term. The latter is typically chosen as a norm of the classifier. The resulting regularized classifier performs better on new data. This phenomenon is often interpreted from a statistical learning theory view: the regularization term restricts the complexity of the classifier, hence the deviation of the testing error and the training error is controlled (see Smola et al., 1998; Evgeniou et al., 2000; Bartlett and Mendelson, 2002; Koltchinskii and Panchenko, 2002; Bartlett et al., 2005, and references therein).

In this paper we consider a different setup, assuming that the training data are generated by the true underlying distribution, but some non-i.i.d. (potentially adversarial) disturbance is then added to the samples we observe. We follow a robust optimization (see El Ghaoui and Lebret, 1997; Ben-Tal and Nemirovski, 1999; Bertsimas and Sim, 2004, and references therein) approach, that is, minimizing the worst possible empirical error under such disturbances. The use of robust optimization in classification is not new (e.g., Shivaswamy et al., 2006; Bhattacharyya et al., 2004b; Lanckriet et al., 2003), in which *box-type* uncertainty sets were considered. Moreover, there has not been an explicit connection to the regularized classifier, although at a high-level it is known that regularization and robust optimization are related (e.g., El Ghaoui and Lebret, 1997; Anthony and Bartlett, 1999). The main contribution in this paper is solving the robust classification problem for a class of non-box-typed uncertainty sets, and providing a linkage between robust classification and the standard regularization scheme of SVMs. In particular, our contributions include the following:

- We solve the robust SVM formulation for a class of non-box-type uncertainty sets. This permits finer control of the adversarial disturbance, restricting it to satisfy aggregate constraints across data points, therefore reducing the possibility of highly correlated disturbance.

- We show that the standard regularized SVM classifier is a special case of our robust classification, thus explicitly relating robustness and regularization. This provides an alternative explanation to the success of regularization, and also suggests new physically motivated ways to construct regularization terms.

- We relate our robust formulation to several probabilistic formulations. We consider a chance-constrained classifier (that is, a classifier with probabilistic constraints on misclassification) and show that our robust formulation can approximate it far less conservatively than previous robust formulations could possibly do. We also consider a Bayesian setup, and show that this can be used to provide a principled means of selecting the regularization coefficient without cross-validation.

- We show that the robustness perspective, stemming from a non-i.i.d. analysis, can be useful in the standard learning (i.i.d.) setup, by using it to prove consistency for standard SVM classification, *without using VC-dimension or stability arguments*. This result implies that generalization ability is a direct result of robustness to local disturbances; it therefore suggests a new justification for good performance, and consequently allows us to construct learning algorithms that generalize well by robustifying non-consistent algorithms.

1.1 Robustness and Regularization

We comment here on the explicit equivalence of robustness and regularization. We briefly explain how this observation is different from previous work and why it is interesting. Previous works on robust classification (e.g., Lanckriet et al., 2003; Bhattacharyya et al., 2004a,b; Shivaswamy et al., 2006; Trafalis and Gilbert, 2007) consider robustifying *regularized* classifications.[1] That is, they propose modifications to standard regularized classifications so that the new formulations are robust to input uncertainty. Furthermore, box-type uncertainty—a setup where the joint uncertainty is the Cartesian product of uncertainty in each input (see Section 2 for detailed formulation)—is considered, which leads to penalty terms on each *constraint* of the resulting formulation. The objective of these works was not to relate robustness and the standard regularization term that appears in the *objective function*. Indeed, research on classifier regularization mainly considers its effect on bounding the complexity of the function class (e.g., Smola et al., 1998; Evgeniou et al., 2000; Bartlett and Mendelson, 2002; Koltchinskii and Panchenko, 2002; Bartlett et al., 2005). Thus, although certain equivalence relationships between robustness and regularization have been established for problems other than classification (El Ghaoui and Lebret, 1997; Ben-Tal and Nemirovski, 1999; Bishop, 1995; Xu et al., 2009), the explicit equivalence between robustness and regularization in the SVM setup is novel.

The connection of robustness and regularization in the SVM context is important for the following reasons. First, it gives an alternative and potentially powerful explanation of the generalization ability of the regularization term. In the standard machine learning view, the regularization term bounds the complexity of the class of classifiers. The robust view of regularization regards the testing samples as a perturbed copy of the training samples. Therefore, when the total perturbation is given or bounded, the regularization term bounds the gap between the classification errors of the SVM on these two sets of samples. In contrast to the standard PAC approach, this bound depends neither on how rich the class of candidate classifiers is, nor on an assumption that all samples are picked in an i.i.d. manner.

Second, this connection suggests novel approaches to designing good classification algorithms, in particular, designing the regularization term. In the PAC structural-risk minimization approach, regularization is chosen to minimize a bound on the generalization error based on the training error and a complexity term. This approach is known to often be too pessimistic (Kearns et al., 1997), especially for problems with more structure. The robust approach offers another avenue. Since both noise and robustness are physical processes, a close investigation of the application and noise characteristics at hand, can provide insights into how to properly robustify, and therefore regularize the classifier. For example, it is known that normalizing the samples so that the variance among all features is roughly the same (a process commonly used to eliminate the scaling freedom of individual features) often leads to good generalization performance. From the robustness perspective, this has the interpretation that the noise is anisotropic (ellipsoidal) rather than spherical, and hence an appropriate robustification must be designed to fit this anisotropy.

We also show that using the robust optimization viewpoint, we obtain some probabilistic results that go beyond the PAC setup. In Section 3 we bound the probability that a noisy training sample is correctly labeled. Such a bound considers the behavior of *corrupted* samples and is hence different from the known PAC bounds. This is helpful when the training samples and the testing samples are

1. Lanckriet et al. (2003) is perhaps the only exception, where a regularization term is added to the covariance estimation rather than to the objective function.

drawn from different distributions, or some adversary manipulates the samples to prevent them from being correctly labeled (e.g., spam senders change their patterns from time to time to avoid being labeled and filtered). Finally, this connection of robustification and regularization also provides us with new proof techniques as well (see Section 5).

We need to point out that there are several different definitions of robustness in the literature. In this paper, as well as the aforementioned robust classification papers, robustness is mainly understood from a Robust Optimization (RO) perspective, where a min-max optimization is performed over all possible disturbances. An alternative interpretation of robustness stems from the rich literature on robust statistics (e.g., Huber, 1981; Hampel et al., 1986; Rousseeuw and Leroy, 1987; Maronna et al., 2006), which studies how an estimator or algorithm behaves under a small perturbation of the statistics model. For example, the *influence function* approach, proposed in Hampel (1974) and Hampel et al. (1986), measures the impact of an infinitesimal amount of contamination of the original distribution on the quantity of interest. Based on this notion of robustness, Christmann and Steinwart (2004) showed that many kernel classification algorithms, including SVM, are robust in the sense of having a finite Influence Function. A similar result for regression algorithms is shown in Christmann and Steinwart (2007) for smooth loss functions, and in Christmann and Van Messem (2008) for non-smooth loss functions where a relaxed version of the Influence Function is applied. In the machine learning literature, another widely used notion closely related to robustness is the *stability*, where an algorithm is required to be robust (in the sense that the output function does not change significantly) under a specific perturbation: deleting one sample from the training set. It is now well known that a stable algorithm such as SVM has desirable generalization properties, and is statistically consistent under mild technical conditions; see for example Bousquet and Elisseeff (2002), Kutin and Niyogi (2002), Poggio et al. (2004) and Mukherjee et al. (2006) for details. One main difference between RO and other robustness notions is that the former is constructive rather than analytical. That is, in contrast to robust statistics or the stability approach that measures the robustness of a *given* algorithm, RO can *robustify* an algorithm: it converts a given algorithm to a robust one. For example, as we show in this paper, the RO version of a naive empirical-error minimization is the well known SVM. As a constructive process, the RO approach also leads to additional flexibility in algorithm design, especially when the nature of the perturbation is known or can be well estimated.

1.2 Structure of the Paper

This paper is organized as follows. In Section 2 we investigate the correlated disturbance case, and show the equivalence between the robust classification and the regularization process. We develop the connections to probabilistic formulations in Section 3. The kernelized version is investigated in Section 4. Finally, in Section 5, we consider the standard statistical learning setup where all samples are i.i.d. draws and provide a novel proof of consistency of SVM based on robustness analysis. The analysis shows that duplicate copies of iid draws tend to be "similar" to each other in the sense that with high probability the total difference is small, and hence robustification that aims to control performance loss for small perturbations can help mitigate overfitting even though no explicit perturbation exists.

1.3 Notation

Capital letters are used to denote matrices, and boldface letters are used to denote column vectors. For a given norm $\|\cdot\|$, we use $\|\cdot\|^*$ to denote its dual norm, that is, $\|\mathbf{z}\|^* \triangleq \sup\{\mathbf{z}^\top \mathbf{x} | \|\mathbf{x}\| \leq 1\}$. For a vector $\mathbf{x}$ and a positive semi-definite matrix C of the same dimension, $\|\mathbf{x}\|_C$ denotes $\sqrt{\mathbf{x}^\top C \mathbf{x}}$. We use δ to denote disturbance affecting the samples. We use superscript r to denote the true value for an uncertain variable, so that δ_i^r is the true (but unknown) noise of the i^{th} sample. The set of non-negative scalars is denoted by $\mathbb{R}^+$. The set of integers from 1 to n is denoted by $[1:n]$.

2. Robust Classification and Regularization

We consider the standard binary classification problem, where we are given a finite number of training samples $\{\mathbf{x}_i, y_i\}_{i=1}^m \subseteq \mathbb{R}^n \times \{-1,+1\}$, and must find a linear classifier, specified by the function $h^{\mathbf{w},b}(\mathbf{x}) = \mathrm{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$. For the standard regularized classifier, the parameters $(\mathbf{w}, b)$ are obtained by solving the following convex optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} : \quad & r(\mathbf{w},b) + \sum_{i=1}^m \xi_i \\ \text{s.t.} : \quad & \xi_i \geq \left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)\right] \\ & \xi_i \geq 0, \end{aligned}$$

where $r(\mathbf{w},b)$ is a regularization term. This is equivalent to

$$\min_{\mathbf{w},b} \left\{ r(\mathbf{w},b) + \sum_{i=1}^m \max\left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b), 0\right] \right\}.$$

Previous robust classification work (Shivaswamy et al., 2006; Bhattacharyya et al., 2004a,b; Bhattacharyya, 2004; Trafalis and Gilbert, 2007) considers the classification problem where the input are subject to (unknown) disturbances $\vec{\delta} = (\delta_1, \ldots, \delta_m)$ and essentially solves the following min-max problem:

$$\min_{\mathbf{w},b} \max_{\vec{\delta} \in \mathcal{N}_{\text{box}}} \left\{ r(\mathbf{w},b) + \sum_{i=1}^m \max\left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i - \delta_i \rangle + b), 0\right] \right\}, \tag{1}$$

for a box-type uncertainty set $\mathcal{N}_{\text{box}}$. That is, let $\mathcal{N}_i$ denote the projection of $\mathcal{N}_{\text{box}}$ onto the δ_i component, then $\mathcal{N}_{\text{box}} = \mathcal{N}_1 \times \cdots \times \mathcal{N}_m$ (note that $\mathcal{N}_i$ need not be a "box"). Effectively, this allows simultaneous worst-case disturbances across many samples, and leads to overly conservative solutions. The goal of this paper is to obtain a robust formulation where the disturbances $\{\delta_i\}$ may be meaningfully taken to be correlated, that is, to solve for a non-box-type $\mathcal{N}$:

$$\min_{\mathbf{w},b} \max_{\vec{\delta} \in \mathcal{N}} \left\{ r(\mathbf{w},b) + \sum_{i=1}^m \max\left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i - \delta_i \rangle + b), 0\right] \right\}. \tag{2}$$

We briefly explain here the four reasons that motivate this "robust to perturbation" setup and in particular the min-max form of (1) and (2). First, it can explicitly incorporate prior problem knowledge of local invariance (e.g., Teo et al., 2008). For example, in vision tasks, a desirable classifier should provide a consistent answer if an input image slightly changes. Second, there are situations where

some adversarial opponents (e.g., spam senders) will manipulate the testing samples to avoid being correctly classified, and the robustness toward such manipulation should be taken into consideration in the training process (e.g., Globerson and Roweis, 2006). Or alternatively, the training samples and the testing samples can be obtained from different processes and hence the standard i.i.d. assumption is violated (e.g., Bi and Zhang, 2004). For example in real-time applications, the newly generated samples are often less accurate due to time constraints. Finally, formulations based on chance-constraints (e.g., Bhattacharyya et al., 2004b; Shivaswamy et al., 2006) are mathematically equivalent to such a min-max formulation.

We define explicitly the correlated disturbance (or uncertainty) which we study below.

Definition 1 *A set $\mathcal{N}_0 \subseteq \mathbb{R}^n$ is called an* Atomic Uncertainty Set *if*

$$\begin{aligned} &(I) \quad \mathbf{0} \in \mathcal{N}_0; \\ &(II) \quad \text{For any } \mathbf{w}_0 \in \mathbb{R}^n : \sup_{\delta \in \mathcal{N}_0} [\mathbf{w}_0^\top \delta] = \sup_{\delta' \in \mathcal{N}_0} [-\mathbf{w}_0^\top \delta'] < +\infty. \end{aligned}$$

We use "sup" here because the maximal value is not necessary attained since $\mathcal{N}_0$ may not be a closed set. The second condition of Atomic Uncertainty set basically says that the uncertainty set is bounded and symmetric. In particular, all norm balls and ellipsoids centered at the origin are atomic uncertainty sets, while an arbitrary polytope might not be an atomic uncertainty set.

Definition 2 *Let $\mathcal{N}_0$ be an atomic uncertainty set. A set $\mathcal{N} \subseteq \mathbb{R}^{n \times m}$ is called a* Sublinear Aggregated Uncertainty Set *of $\mathcal{N}_0$, if*

$$\mathcal{N}^- \subseteq \mathcal{N} \subseteq \mathcal{N}^+,$$

$$\text{where:} \quad \mathcal{N}^- \triangleq \bigcup_{t=1}^{m} \mathcal{N}_t^-; \qquad \mathcal{N}_t^- \triangleq \{(\delta_1, \cdots, \delta_m) | \delta_t \in \mathcal{N}_0;\ \delta_{i \neq t} = \mathbf{0}\}.$$

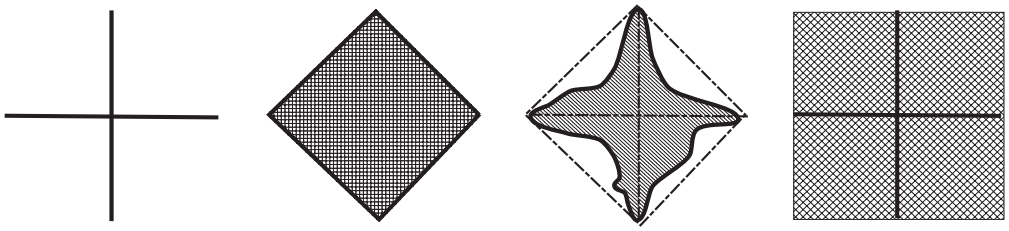
$$\mathcal{N}^+ \triangleq \{(\alpha_1 \delta_1, \cdots, \alpha_m \delta_m) | \sum_{i=1}^{m} \alpha_i = 1;\ \alpha_i \geq 0,\ \delta_i \in \mathcal{N}_0, i = 1, \cdots, m\}.$$

The Sublinear Aggregated Uncertainty definition models the case where the disturbances on each sample are treated identically, but their aggregate behavior across multiple samples is controlled. Some interesting examples include

$$\begin{aligned} &(1) \quad \{(\delta_1, \cdots, \delta_m) | \sum_{i=1}^{m} \|\delta_i\| \leq c\}; \\ &(2) \quad \{(\delta_1, \cdots, \delta_m) | \exists t \in [1:m];\ \|\delta_t\| \leq c;\ \delta_i = \mathbf{0}, \forall i \neq t\}; \\ &(3) \quad \{(\delta_1, \cdots, \delta_m) | \sum_{i=1}^{m} \sqrt{c\|\delta_i\|} \leq c\}. \end{aligned}$$

All these examples have the same atomic uncertainty set $\mathcal{N}_0 = \left\{\delta \middle| \|\delta\| \leq c\right\}$. Figure 1 provides an illustration of a sublinear aggregated uncertainty set for $n = 1$ and $m = 2$, that is, the training set consists of two univariate samples.

The following theorem is the main result of this section, which reveals that standard norm regularized SVM is the solution of a (non-regularized) robust optimization. It is a special case of Proposition 4 by taking $\mathcal{N}_0$ as the dual-norm ball $\{\delta | \|\delta\|^* \leq c\}$ for an arbitrary norm $\|\cdot\|$ and $r(\mathbf{w}, b) \equiv 0$.



a. $\mathcal{N}^-$ b. $\mathcal{N}^+$ c. $\mathcal{N}$ d. Box uncertainty

Figure 1: Illustration of a Sublinear Aggregated Uncertainty Set $\mathcal{N}$.

Theorem 3 *Let* $\mathcal{T} \triangleq \left\{(\delta_1, \cdots \delta_m) | \sum_{i=1}^m \|\delta_i\|^* \leq c\right\}$. *Suppose that the training sample* $\{\mathbf{x}_i, y_i\}_{i=1}^m$ *are non-separable. Then the following two optimization problems on* $(\mathbf{w}, b)$ *are equivalent*[2]

$$\begin{aligned} \min: \quad & \max_{(\delta_1,\cdots,\delta_m)\in\mathcal{T}} \sum_{i=1}^m \max\left[1 - y_i\big(\langle \mathbf{w}, \mathbf{x}_i - \delta_i\rangle + b\big), 0\right], \\ \min: \quad & c\|\mathbf{w}\| + \sum_{i=1}^m \max\left[1 - y_i\big(\langle \mathbf{w}, \mathbf{x}_i\rangle + b\big), 0\right]. \end{aligned} \tag{3}$$

Proposition 4 *Assume* $\{\mathbf{x}_i, y_i\}_{i=1}^m$ *are non-separable,* $r(\cdot) : \mathbb{R}^{n+1} \to \mathbb{R}$ *is an arbitrary function,* $\mathcal{N}$ *is a Sublinear Aggregated Uncertainty set with corresponding atomic uncertainty set* $\mathcal{N}_0$. *Then the following min-max problem*

$$\min_{\mathbf{w},b} \sup_{(\delta_1,\cdots,\delta_m)\in\mathcal{N}} \left\{ r(\mathbf{w}, b) + \sum_{i=1}^m \max\left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i - \delta_i\rangle + b), 0\right]\right\} \tag{4}$$

is equivalent to the following optimization problem on $\mathbf{w}, b, \xi$*:*

$$\begin{aligned} \min: \quad & r(\mathbf{w}, b) + \sup_{\delta\in\mathcal{N}_0} (\mathbf{w}^\top \delta) + \sum_{i=1}^m \xi_i, \\ \text{s.t.}: \quad & \xi_i \geq 1 - [y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)], \quad i = 1, \ldots, m; \\ & \xi_i \geq 0, \quad i = 1, \ldots, m. \end{aligned} \tag{5}$$

Furthermore, the minimization of Problem (5) is attainable when $r(\cdot, \cdot)$ *is lower semi-continuous.*

Proof Define:

$$v(\mathbf{w}, b) \quad \triangleq \quad \sup_{\delta\in\mathcal{N}_0} (\mathbf{w}^\top \delta) + \sum_{i=1}^m \max\left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b), 0\right].$$

2. The optimization equivalence for the linear case was observed independently by Bertsimas and Fertis (2008).

Recall that $\mathcal{N}^- \subseteq \mathcal{N} \subseteq N^+$ by definition. Hence, fixing any $(\hat{\mathbf{w}}, \hat{b}) \in \mathbb{R}^{n+1}$, the following inequalities hold:

$$\begin{aligned}
&\sup_{(\delta_1,\cdots,\delta_m)\in\mathcal{N}^-} \sum_{i=1}^{m} \max\left[1 - y_i(\langle \hat{\mathbf{w}}, \mathbf{x}_i - \delta_i\rangle + \hat{b}), 0\right] \\
\leq &\sup_{(\delta_1,\cdots,\delta_m)\in\mathcal{N}} \sum_{i=1}^{m} \max\left[1 - y_i(\langle \hat{\mathbf{w}}, \mathbf{x}_i - \delta_i\rangle + \hat{b}), 0\right] \\
\leq &\sup_{(\delta_1,\cdots,\delta_m)\in\mathcal{N}^+} \sum_{i=1}^{m} \max\left[1 - y_i(\langle \hat{\mathbf{w}}, \mathbf{x}_i - \delta_i\rangle + \hat{b}), 0\right].
\end{aligned}$$

To prove the theorem, we first show that $v(\hat{\mathbf{w}}, \hat{b})$ is no larger than the leftmost expression and then show $v(\hat{\mathbf{w}}, \hat{b})$ is no smaller than the rightmost expression.

Step 1: We prove that

$$v(\hat{\mathbf{w}}, \hat{b}) \leq \sup_{(\delta_1,\cdots,\delta_m)\in\mathcal{N}^-} \sum_{i=1}^{m} \max\left[1 - y_i(\langle \hat{\mathbf{w}}, \mathbf{x}_i - \delta_i\rangle + \hat{b}), 0\right]. \tag{6}$$

Since the samples $\{\mathbf{x}_i, y_i\}_{i=1}^m$ are not separable, there exists $t \in [1:m]$ such that

$$y_t(\langle \hat{\mathbf{w}}, \mathbf{x}_t\rangle + \hat{b}) < 0. \tag{7}$$

Hence,

$$\begin{aligned}
&\sup_{(\delta_1,\cdots,\delta_m)\in\mathcal{N}_t^-} \sum_{i=1}^{m} \max\left[1 - y_i(\langle \hat{\mathbf{w}}, \mathbf{x}_i - \delta_i\rangle + \hat{b}), 0\right] \\
&= \sum_{i\neq t} \max\left[1 - y_i(\langle \hat{\mathbf{w}}, \mathbf{x}_i\rangle + \hat{b}), 0\right] + \sup_{\delta_t\in\mathcal{N}_0} \max\left[1 - y_t(\langle \hat{\mathbf{w}}, \mathbf{x}_t - \delta_t\rangle + \hat{b}), 0\right] \\
&= \sum_{i\neq t} \max\left[1 - y_i(\langle \hat{\mathbf{w}}, \mathbf{x}_i\rangle + \hat{b}), 0\right] + \max\left[1 - y_t(\langle \hat{\mathbf{w}}, \mathbf{x}_t\rangle + \hat{b}) + \sup_{\delta_t\in\mathcal{N}_0} (y_t \hat{\mathbf{w}}^\top \delta_t), 0\right] \\
&= \sum_{i\neq t} \max\left[1 - y_i(\langle \hat{\mathbf{w}}, \mathbf{x}_i\rangle + \hat{b}), 0\right] + \max\left[1 - y_t(\langle \hat{\mathbf{w}}, \mathbf{x}_t\rangle + \hat{b}), 0\right] + \sup_{\delta_t\in\mathcal{N}_0} (y_t \hat{\mathbf{w}}^\top \delta_t) \\
&= \sup_{\delta\in\mathcal{N}_0} (\hat{\mathbf{w}}^\top \delta) + \sum_{i=1}^{m} \max\left[1 - y_i(\langle \hat{\mathbf{w}}, \mathbf{x}_i\rangle + \hat{b}), 0\right] = v(\hat{\mathbf{w}}, \hat{b}).
\end{aligned}$$

The third equality holds because of Inequality (7) and $\sup_{\delta_t\in\mathcal{N}_0}(y_t \hat{\mathbf{w}}^\top \delta_t)$ being non-negative (recall $\mathbf{0} \in \mathcal{N}_0$). Since $\mathcal{N}_t^- \subseteq \mathcal{N}^-$, Inequality (6) follows.

Step 2: Next we prove that

$$\sup_{(\delta_1,\cdots,\delta_m)\in\mathcal{N}^+} \sum_{i=1}^{m} \max\left[1 - y_i(\langle \hat{\mathbf{w}}, \mathbf{x}_i - \delta_i\rangle + \hat{b}), 0\right] \leq v(\hat{\mathbf{w}}, \hat{b}). \tag{8}$$

Notice that by the definition of $\mathcal{N}^+$ we have

$$\begin{aligned}
&\sup_{(\delta_1,\cdots,\delta_m)\in\mathcal{N}^+}\sum_{i=1}^m \max\left[1-y_i(\langle\hat{\mathbf{w}},\mathbf{x}_i-\delta_i\rangle+\hat{b}),0\right]\\
=&\sup_{\sum_{i=1}^m\alpha_i=1;\alpha_i\geq 0;\hat{\delta}_i\in\mathcal{N}_0}\sum_{i=1}^m \max\left[1-y_i(\langle\hat{\mathbf{w}},\mathbf{x}_i-\alpha_i\hat{\delta}_i\rangle+\hat{b}),0\right] \qquad (9)\\
=&\sup_{\sum_{i=1}^m\alpha_i=1;\alpha_i\geq 0;}\sum_{i=1}^m \max\Big[\sup_{\hat{\delta}_i\in\mathcal{N}_0}\left(1-y_i(\langle\hat{\mathbf{w}},\mathbf{x}_i-\alpha_i\hat{\delta}_i\rangle+\hat{b})\right),0\Big].
\end{aligned}$$

Now, for any $i\in[1:m]$, the following holds,

$$\begin{aligned}
&\max\Big[\sup_{\hat{\delta}_i\in\mathcal{N}_0}\left(1-y_i(\langle\hat{\mathbf{w}},\mathbf{x}_i-\alpha_i\hat{\delta}_i\rangle+\hat{b})\right),0\Big]\\
=&\max\left[1-y_i(\langle\hat{\mathbf{w}},\mathbf{x}_i\rangle+\hat{b})+\alpha_i\sup_{\hat{\delta}_i\in\mathcal{N}_0}(\hat{\mathbf{w}}^\top\hat{\delta}_i),0\right]\\
\leq&\max\left[1-y_i(\langle\hat{\mathbf{w}},\mathbf{x}_i\rangle+\hat{b}),0\right]+\alpha_i\sup_{\hat{\delta}_i\in\mathcal{N}_0}(\hat{\mathbf{w}}^\top\hat{\delta}_i).
\end{aligned}$$

Therefore, Equation (9) is upper bounded by

$$\begin{aligned}
&\sum_{i=1}^m\max\left[1-y_i(\langle\hat{\mathbf{w}},\mathbf{x}_i\rangle+\hat{b}),0\right]+\sup_{\sum_{i=1}^m\alpha_i=1;\alpha_i\geq 0;}\sum_{i=1}^m\alpha_i\sup_{\hat{\delta}_i\in\mathcal{N}_0}(\hat{\mathbf{w}}^\top\hat{\delta}_i)\\
=&\sup_{\delta\in\mathcal{N}_0}(\hat{\mathbf{w}}^\top\delta)+\sum_{i=1}^m\max\left[1-y_i(\langle\hat{\mathbf{w}},\mathbf{x}_i\rangle+\hat{b}),0\right]=v(\hat{\mathbf{w}},\hat{b}),
\end{aligned}$$

hence Inequality (8) holds.

Step 3: Combining the two steps and adding $r(\mathbf{w},b)$ on both sides leads to: $\forall(\mathbf{w},b)\in\mathbb{R}^{n+1}$,

$$\sup_{(\delta_1,\cdots,\delta_m)\in\mathcal{N}}\sum_{i=1}^m\max\left[1-y_i(\langle\mathbf{w},\mathbf{x}_i-\delta_i\rangle+b),0\right]+r(\mathbf{w},b)=v(\mathbf{w},b)+r(\mathbf{w},\mathbf{b}).$$

Taking the infimum on both sides establishes the equivalence of Problem (4) and Problem (5). Observe that $\sup_{\delta\in\mathcal{N}_0}\mathbf{w}^\top\delta$ is a supremum over a class of affine functions, and hence is lower semi-continuous. Therefore $v(\cdot,\cdot)$ is also lower semi-continuous. Thus the minimum can be achieved for Problem (5), and Problem (4) by equivalence, when $r(\cdot)$ is lower semi-continuous. ■

Before concluding this section we briefly comment on the meaning of Theorem 3 and Proposition 4. On one hand, they explain the widely known fact that the regularized classifier tends to be more robust (see for example, Christmann and Steinwart, 2004, 2007; Christmann and Van Messem, 2008; Trafalis and Gilbert, 2007). On the other hand, this observation also suggests that the appropriate way to regularize should come from a disturbance-robustness perspective. The above equivalence implies that standard regularization essentially assumes that the disturbance is spherical; if this is not true, robustness may yield a better regularization-like algorithm. To find a more effective regularization term, a closer investigation of the data variation is desirable, particularly if some a-priori knowledge of the data-variation is known. For example, consider an image

classification problem. Suppose it is known that these pictures are taken under significantly varying background light. Therefore, for a given sample (picture), the perturbation on each feature (pixel) is large. However, the perturbations across different features are almost identical since they are under the same background light. This can be represented by the following Atomic uncertainty set

$$\mathcal{N}_0 = \{\delta | \|\delta\|_2 \le c_1,\ \|\delta - (\frac{1}{n}\sum_{t=1}^{n}\delta_t)\mathbf{1}\|_2 \le c_2\},$$

where $c_2 \ll c_1$. By Proposition 4, this leads to the following regularization term

$$\begin{aligned} f(\mathbf{w}) = \ & \max : \mathbf{w}^\top \delta \\ & \text{s.t.} \ \|\delta\|_2 \le c_1 \\ & \quad \|(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)\delta\|_2 \le c_2. \end{aligned}$$

Notice this is a second order cone programming which has a dual form

$$\begin{aligned} \min : \ & c_1 v_1 + c_2 v_2 \\ \text{s.t.} \ & \mathbf{u}_1 + (I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)\mathbf{u}_2 = \mathbf{w} \\ & \|\mathbf{u}_i\|_2 \le v_i,\ i = 1, 2. \end{aligned}$$

Substituting it to (5), the resulting classification problem is a second order cone program, which can be efficiently solved (Boyd and Vandenberghe, 2004).

3. Probabilistic Interpretations

Although Problem (4) is formulated without any probabilistic assumptions, in this section, we briefly explain two approaches to construct the uncertainty set and equivalently tune the regularization parameter c based on probabilistic information.

The first approach is to use Problem (4) to approximate an upper bound for a chance-constrained classifier. Suppose the disturbance $(\delta_1^r, \cdots \delta_m^r)$ follows a joint probability measure μ. Then the chance-constrained classifier is given by the following minimization problem given a confidence level $\eta \in [0, 1]$,

$$\begin{aligned} \min_{\mathbf{w},b,l} : \quad & l \\ \text{s.t.} : \quad & \mu\Big\{ \sum_{i=1}^{m} \max\left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i - \delta_i^r \rangle + b), 0\right] \le l \Big\} \ge 1 - \eta. \end{aligned} \tag{10}$$

The formulations in Shivaswamy et al. (2006), Lanckriet et al. (2003) and Bhattacharyya et al. (2004a) assume uncorrelated noise and require all constraints to be satisfied with high probability *simultaneously*. They find a vector $[\xi_1, \cdots, \xi_m]^\top$ where each ξ_i is the η-quantile of the hinge-loss for sample $\mathbf{x}_i^r$. In contrast, our formulation above minimizes the η-quantile of the average (or equivalently the sum of) empirical error. When controlling this average quantity is of more interest, the box-type noise formulation will be overly conservative.

Problem (10) is generally intractable. However, we can approximate it as follows. Let

$$c^* \triangleq \inf\{\alpha | \mu(\sum_i \|\delta_i\|^* \le \alpha) \ge 1 - \eta\}.$$

Notice that c^* is easily simulated given μ. Then for any $(\mathbf{w}, b)$, with probability no less than $1-\eta$, the following holds,

$$\sum_{i=1}^{m} \max\left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i - \delta_i^r \rangle + b), 0\right]$$
$$\leq \max_{\sum_i \|\delta_i\|^* \leq c^*} \sum_{i=1}^{m} \max\left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i - \delta_i \rangle + b), 0\right].$$

Thus (10) is upper bounded by (3) with $c = c^*$. This gives an additional probabilistic robustness property of the standard regularized classifier. Notice that following a similar approach but with the constraint-wise robust setup, that is, the box uncertainty set, would lead to considerably more pessimistic approximations of the chance constraint.

The second approach considers a Bayesian setup. Suppose the total disturbance $c^r \triangleq \sum_{i=1}^m \|\delta_i^r\|^*$ follows a prior distribution $\rho(\cdot)$. This can model for example the case that the training sample set is a mixture of several data sets where the disturbance magnitude of each set is known. Such a setup leads to the following classifier which minimizes the Bayesian (robust) error:

$$\min_{\mathbf{w},b}: \quad \int \left\{ \max_{\sum \|\delta_i\|^* \leq c} \sum_{i=1}^{m} \max\left[1 - y_i\big(\langle \mathbf{w}, \mathbf{x}_i - \delta_i \rangle + b\big), 0\right] \right\} d\rho(c). \tag{11}$$

By Theorem 3, the Bayes classifier (11) is equivalent to

$$\min_{\mathbf{w},b}: \quad \int \left\{ c\|\mathbf{w}\| + \sum_{i=1}^{m} \max\left[1 - y_i\big(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\big), 0\right] \right\} d\rho(c),$$

which can be further simplified as

$$\min_{\mathbf{w},b}: \quad \overline{c}\|\mathbf{w}\| + \sum_{i=1}^{m} \max\left[1 - y_i\big(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\big), 0\right],$$

where $\overline{c} \triangleq \int c\, d\rho(c)$. This thus provides us a justifiable parameter tuning method different from cross validation: simply using the expected value of c^r. We note that it is the equivalence of Theorem 3 that makes this possible, since it is difficult to imagine a setting where one would have a prior on regularization coefficients.

4. Kernelization

The previous results can be easily generalized to the kernelized setting, which we discuss in detail in this section. In particular, similar to the linear classification case, we give a new interpretation of the standard kernelized SVM as the min-max empirical hinge-loss solution, where the disturbance is assumed to lie in the feature space. We then relate this to the (more intuitively appealing) setup where the disturbance lies in the sample space. We use this relationship in Section 5 to prove a consistency result for kernelized SVMs.

The kernelized SVM formulation considers a linear classifier in the feature space $\mathcal{H}$, a Hilbert space containing the range of some feature mapping $\Phi(\cdot)$. The standard formulation is as follows,

$$\begin{aligned} \min_{\mathbf{w},b}: \quad & r(\mathbf{w}, b) + \sum_{i=1}^{m} \xi_i \\ \text{s.t.}: \quad & \xi_i \geq \left[1 - y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b)\right], \\ & \xi_i \geq 0\,. \end{aligned}$$

It has been proved in Schölkopf and Smola (2002) that if we take $f(\langle \mathbf{w}, \mathbf{w} \rangle)$ for some increasing function $f(\cdot)$ as the regularization term $r(\mathbf{w}, b)$, then the optimal solution has a representation $\mathbf{w}^* = \sum_{i=1}^m \alpha_i \Phi(\mathbf{x}_i)$, which can further be solved without knowing explicitly the feature mapping, but by evaluating a kernel function $k(\mathbf{x}, \mathbf{x}') \triangleq \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$ only. This is the well-known "kernel trick".

The definitions of Atomic Uncertainty Set and Sublinear Aggregated Uncertainty Set in the feature space are identical to Definition 1 and 2, with $\mathbb{R}^n$ replaced by $\mathcal{H}$. The following theorem is a feature-space counterpart of Proposition 4. The proof follows from a similar argument to Proposition 4, that is, for any fixed $(\mathbf{w}, b)$ the worst-case empirical error equals the empirical error plus a penalty term $\sup_{\delta \in \mathcal{N}_0} \big(\langle \mathbf{w}, \delta \rangle \big)$, and hence the details are omitted.

Theorem 5 *Assume $\{\Phi(\mathbf{x}_i), y_i\}_{i=1}^m$ are not linearly separable, $r(\cdot) : \mathcal{H} \times \mathbb{R} \to \mathbb{R}$ is an arbitrary function, $\mathcal{N} \subseteq \mathcal{H}^m$ is a Sublinear Aggregated Uncertainty set with corresponding atomic uncertainty set $\mathcal{N}_0 \subseteq \mathcal{H}$. Then the following min-max problem*

$$\min_{\mathbf{w}, b} \sup_{(\delta_1, \cdots, \delta_m) \in \mathcal{N}} \left\{ r(\mathbf{w}, b) + \sum_{i=1}^m \max \left[1 - y_i (\langle \mathbf{w}, \Phi(\mathbf{x}_i) - \delta_i \rangle + b), 0 \right] \right\}$$

is equivalent to

$$\begin{aligned} \min: \quad & r(\mathbf{w}, b) + \sup_{\delta \in \mathcal{N}_0} \big(\langle \mathbf{w}, \delta \rangle \big) + \sum_{i=1}^m \xi_i, \\ \text{s.t.}: \quad & \xi_i \geq 1 - y_i \big(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b \big), \quad i = 1, \cdots, m; \\ & \xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned} \tag{12}$$

Furthermore, the minimization of Problem (12) is attainable when $r(\cdot, \cdot)$ is lower semi-continuous.

For some widely used feature mappings (e.g., RKHS of a Gaussian kernel), $\{\Phi(\mathbf{x}_i), y_i\}_{i=1}^m$ are always separable. In this case, the worst-case empirical error may not be equal to the empirical error plus a penalty term $\sup_{\delta \in \mathcal{N}_0} \big(\langle \mathbf{w}, \delta \rangle \big)$. However, it is easy to show that for any $(\mathbf{w}, b)$, the latter is an upper bound of the former.

The next corollary is the feature-space counterpart of Theorem 3, where $\| \cdot \|_{\mathcal{H}}$ stands for the RKHS norm, that is, for $\mathbf{z} \in \mathcal{H}$, $\|\mathbf{z}\|_{\mathcal{H}} = \sqrt{\langle \mathbf{z}, \mathbf{z} \rangle}$. Noticing that the RKHS norm is self dual, we find that the proof is identical to that of Theorem 3, and hence omit it.

Corollary 6 *Let $\mathcal{T}_{\mathcal{H}} \triangleq \left\{ (\delta_1, \cdots \delta_m) \middle| \sum_{i=1}^m \|\delta_i\|_{\mathcal{H}} \leq c \right\}$. If $\{\Phi(\mathbf{x}_i), y_i\}_{i=1}^m$ are non-separable, then the following two optimization problems on $(\mathbf{w}, b)$ are equivalent*

$$\begin{aligned} \min: \quad & \max_{(\delta_1, \cdots, \delta_m) \in \mathcal{T}_{\mathcal{H}}} \sum_{i=1}^m \max \big[1 - y_i \big(\langle \mathbf{w}, \Phi(\mathbf{x}_i) - \delta_i \rangle + b \big), 0 \big], \\ \min: \quad & c \|\mathbf{w}\|_{\mathcal{H}} + \sum_{i=1}^m \max \big[1 - y_i \big(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b \big), 0 \big]. \end{aligned} \tag{13}$$

Equation (13) is a variant form of the standard SVM that has a squared RKHS norm regularization term, and it can be shown that the two formulations are equivalent up to changing of tradeoff parameter c, since both the empirical hinge-loss and the RKHS norm are convex. Therefore, Corollary 6

essentially means that the standard kernelized SVM is implicitly a robust classifier (without regularization) with disturbance in the feature-space, and the sum of the magnitude of the disturbance is bounded.

Disturbance in the feature-space is less intuitive than disturbance in the sample space, and the next lemma relates these two different notions.

Lemma 7 *Suppose there exists* $\mathcal{X} \subseteq \mathbb{R}^n$, $\rho > 0$, *and a continuous non-decreasing function* $f : \mathbb{R}^+ \to \mathbb{R}^+$ *satisfying* $f(0) = 0$, *such that*

$$k(\mathbf{x},\mathbf{x}) + k(\mathbf{x}',\mathbf{x}') - 2k(\mathbf{x},\mathbf{x}') \leq f(\|\mathbf{x}-\mathbf{x}'\|_2^2), \quad \forall \mathbf{x},\mathbf{x}' \in \mathcal{X}, \|\mathbf{x}-\mathbf{x}'\|_2 \leq \rho$$

then

$$\|\Phi(\hat{\mathbf{x}}+\delta) - \Phi(\hat{\mathbf{x}})\|_{\mathcal{H}} \leq \sqrt{f(\|\delta\|_2^2)}, \quad \forall \|\delta\|_2 \leq \rho,\ \hat{\mathbf{x}}, \hat{\mathbf{x}}+\delta \in \mathcal{X}.$$

In the appendix, we prove a result that provides a tighter relationship between disturbance in the feature space and disturbance in the sample space, for RBF kernels.

Proof Expanding the RKHS norm yields

$$\begin{aligned}
&\|\Phi(\hat{\mathbf{x}}+\delta) - \Phi(\hat{\mathbf{x}})\|_{\mathcal{H}} \\
=&\sqrt{\langle \Phi(\hat{\mathbf{x}}+\delta) - \Phi(\hat{\mathbf{x}}),\, \Phi(\hat{\mathbf{x}}+\delta) - \Phi(\hat{\mathbf{x}}) \rangle} \\
=&\sqrt{\langle \Phi(\hat{\mathbf{x}}+\delta),\, \Phi(\hat{\mathbf{x}}+\delta) \rangle + \langle \Phi(\hat{\mathbf{x}}),\, \Phi(\hat{\mathbf{x}}) \rangle - 2\langle \Phi(\hat{\mathbf{x}}+\delta),\, \Phi(\hat{\mathbf{x}}) \rangle} \\
=&\sqrt{k\big(\hat{\mathbf{x}}+\delta, \hat{\mathbf{x}}+\delta\big) + k\big(\hat{\mathbf{x}}, \hat{\mathbf{x}}\big) - 2k\big(\hat{\mathbf{x}}+\delta,\ \hat{\mathbf{x}}\big)} \\
\leq&\sqrt{f(\|\hat{\mathbf{x}}+\delta-\hat{\mathbf{x}}\|_2^2)} = \sqrt{f(\|\delta\|_2^2)},
\end{aligned}$$

where the inequality follows from the assumption. ■

Lemma 7 essentially says that under certain conditions, robustness in the feature space is a stronger requirement that robustness in the sample space. Therefore, a classifier that achieves robustness in the feature space (the SVM for example) also achieves robustness in the sample space. Notice that the condition of Lemma 7 is rather weak. In particular, it holds for any continuous $k(\cdot,\cdot)$ and bounded $\mathcal{X}$.

In the next section we consider a more foundational property of robustness in the sample space: we show that a classifier that is robust in the sample space is asymptotically consistent. As a consequence of this result for linear classifiers, the above results imply the consistency for a broad class of kernelized SVMs.

5. Consistency of Regularization

In this section we explore a fundamental connection between learning and robustness, by using robustness properties to re-prove the statistical consistency of the linear classifier, and then the kernelized SVM. Indeed, our proof mirrors the consistency proof found in Steinwart (2005), with the key difference that *we replace metric entropy, VC-dimension, and stability conditions used there, with a robustness condition*.

Thus far we have considered the setup where the training-samples are corrupted by certain set-inclusive disturbances. We now turn to the standard statistical learning setup, by assuming that all

training samples and testing samples are generated i.i.d. according to a (unknown) probability $\mathbb{P}$, that is, there does not exist explicit disturbance.

Let $\mathcal{X} \subseteq \mathbb{R}^n$ be bounded, and suppose the training samples $(\mathbf{x}_i, y_i)_{i=1}^{\infty}$ are generated i.i.d. according to an unknown distribution $\mathbb{P}$ supported by $\mathcal{X} \times \{-1, +1\}$. The next theorem shows that our robust classifier setup and equivalently regularized SVM asymptotically minimizes an upper-bound of the expected classification error and hinge loss.

Theorem 8 *Denote* $K \triangleq \max_{x \in \mathcal{X}} \|x\|_2$. *Then there exists a random sequence* $\{\gamma_{m,c}\}$ *such that:*

1. $\forall c > 0$, $\lim_{m\to\infty} \gamma_{m,c} = 0$ *almost surely, and the convergence is uniform in* $\mathbb{P}$*;*

2. *the following bounds on the Bayes loss and the hinge loss hold uniformly for all* $(\mathbf{w}, b)$*:*

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathbb{P}}(\mathbf{1}_{y \neq sgn(\langle \mathbf{w}, \mathbf{x}\rangle + b)}) \leq \gamma_{m,c} + c\|\mathbf{w}\|_2 + \frac{1}{m}\sum_{i=1}^{m} \max\left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b), 0\right];$$

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathbb{P}}\Big(\max(1 - y(\langle \mathbf{w}, \mathbf{x}\rangle + b), 0)\Big) \leq$$

$$\gamma_{m,c}(1 + K\|\mathbf{w}\|_2 + |b|) + c\|\mathbf{w}\|_2 + \frac{1}{m}\sum_{i=1}^{m} \max\left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b), 0\right].$$

Proof We briefly explain the basic idea of the proof before going to the technical details. We consider the testing sample set as a perturbed copy of the training sample set, and measure the magnitude of the perturbation. For testing samples that have "small" perturbations, $c\|\mathbf{w}\|_2 + \frac{1}{m}\sum_{i=1}^{m} \max\left[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b), 0\right]$ upper-bounds their total loss by Theorem 3. Therefore, we only need to show that the ratio of testing samples having "large" perturbations diminishes to prove the theorem.

Now we present the detailed proof. Given a $c > 0$, we call a testing sample $(\mathbf{x}', y')$ and a training sample $(\mathbf{x}, y)$ a *sample pair* if $y = y'$ and $\|\mathbf{x} - \mathbf{x}'\|_2 \leq c$. We say a set of training samples and a set of testing samples form l pairings if there exist l sample pairs with no data reused. Given m training samples and m testing samples, we use $M_{m,c}$ to denote the largest number of pairings. To prove this theorem, we need to establish the following lemma.

Lemma 9 *Given a* $c > 0$, $M_{m,c}/m \to 1$ *almost surely as* $m \to +\infty$, *uniformly w.r.t.* $\mathbb{P}$.

Proof We make a partition of $\mathcal{X} \times \{-1, +1\} = \bigcup_{t=1}^{T_c} \mathcal{X}_t$ such that $\mathcal{X}_t$ either has the form $[\alpha_1, \alpha_1 + c/\sqrt{n}) \times [\alpha_2, \alpha_2 + c/\sqrt{n}) \cdots \times [\alpha_n, \alpha_n + c/\sqrt{n}) \times \{+1\}$ or $[\alpha_1, \alpha_1 + c/\sqrt{n}) \times [\alpha_2, \alpha_2 + c/\sqrt{n}) \cdots \times [\alpha_n, \alpha_n + c/\sqrt{n}) \times \{-1\}$ (recall n is the dimension of $\mathcal{X}$). That is, each partition is the Cartesian product of a rectangular cell in $\mathcal{X}$ and a singleton in $\{-1, +1\}$. Notice that if a training sample and a testing sample fall into $\mathcal{X}_t$, they can form a pairing.

Let N_t^{tr} and N_t^{te} be the number of training samples and testing samples falling in the t^{th} set, respectively. Thus, $(N_1^{tr}, \cdots, N_{T_c}^{tr})$ and $(N_1^{te}, \cdots, N_{T_c}^{te})$ are multinomially distributed random vectors following a same distribution. Notice that for a multinomially distributed random vector $(N_1, \cdots, N_k)$ with parameter m and $(p_1, \cdots, p_k)$, the following holds (Bretegnolle-Huber-Carol inequality, see for example Proposition A6.6 of van der Vaart and Wellner, 2000). For any $\lambda > 0$,

$$\mathbb{P}\Big(\sum_{i=1}^{k} \big|N_i - mp_i\big|) \geq 2\sqrt{m}\lambda\Big) \leq 2^k \exp(-2\lambda^2).$$

Hence we have

$$\begin{aligned}
& \mathbb{P}\Big(\sum_{t=1}^{T_c}\big|N_t^{tr}-N_t^{te}\big|\geq 4\sqrt{m}\lambda\Big)\leq 2^{T_c+1}\exp(-2\lambda^2),\\
\Longrightarrow\quad & \mathbb{P}\Big(\frac{1}{m}\sum_{t=1}^{T_c}\big|N_t^{tr}-N_t^{te}\big|\geq \lambda\Big)\leq 2^{T_c+1}\exp(\frac{-m\lambda^2}{8}),\\
\Longrightarrow\quad & \mathbb{P}\Big(M_{m,c}/m\leq 1-\lambda\Big)\leq 2^{T_c+1}\exp(\frac{-m\lambda^2}{8}). \qquad (14)
\end{aligned}$$

Observe that $\sum_{m=1}^{\infty}2^{T_c+1}\exp(\frac{-m\lambda^2}{8})<+\infty$, hence by the Borel-Cantelli Lemma (see, for example, Durrett, 2004), with probability one the event $\{M_{m,c}/m\leq 1-\lambda\}$ only occurs finitely often as $m\to\infty$. That is, $\liminf_m M_{m,c}/m\geq 1-\lambda$ almost surely. Since λ can be arbitrarily close to zero, $M_{m,c}/m\to 1$ almost surely. Observe that this convergence is uniform in $\mathbb{P}$, since T_c only depends on $\mathcal{X}$. ■

Now we proceed to prove the theorem. Given m training samples and m testing samples with $M_{m,c}$ sample pairs, we notice that for these paired samples, both the total testing error and the total testing hinge-loss is upper bounded by

$$\begin{aligned}
&\max_{(\delta_1,\cdots,\delta_m)\in\mathcal{N}_0\times\cdots\times\mathcal{N}_0}\sum_{i=1}^{m}\max\big[1-y_i\big(\langle\mathbf{w},\mathbf{x}_i-\delta_i\rangle+b\big),0\big]\\
\leq& cm\|\mathbf{w}\|_2+\sum_{i=1}^{m}\max\big[1-y_i(\langle\mathbf{w},\mathbf{x}_i\rangle+b),0\big],
\end{aligned}$$

where $\mathcal{N}_0=\{\delta\,|\,\|\delta\|\leq c\}$. Hence the total classification error of the m testing samples can be upper bounded by

$$(m-M_{m,c})+cm\|\mathbf{w}\|_2+\sum_{i=1}^{m}\max\big[1-y_i(\langle\mathbf{w},\mathbf{x}_i\rangle+b),0\big],$$

and since

$$\max_{\mathbf{x}\in\mathcal{X}}(1-y(\langle\mathbf{w},\mathbf{x}\rangle))\leq\max_{\mathbf{x}\in\mathcal{X}}\Big\{1+|b|+\sqrt{\langle\mathbf{x},\mathbf{x}\rangle\cdot\langle\mathbf{w},\mathbf{w}\rangle}\Big\}=1+|b|+K\|\mathbf{w}\|_2,$$

the accumulated hinge-loss of the total m testing samples is upper bounded by

$$(m-M_{m,c})(1+K\|\mathbf{w}\|_2+|b|)+cm\|\mathbf{w}\|_2+\sum_{i=1}^{m}\max\big[1-y_i(\langle\mathbf{w},\mathbf{x}_i\rangle+b),0\big].$$

Therefore, the average testing error is upper bounded by

$$1-M_{m,c}/m+c\|\mathbf{w}\|_2+\frac{1}{m}\sum_{i=1}^{n}\max\big[1-y_i(\langle\mathbf{w},\mathbf{x}_i\rangle+b),0\big],$$

and the average hinge loss is upper bounded by

$$(1-M_{m,c}/m)(1+K\|\mathbf{w}\|_2+|b|)+c\|\mathbf{w}\|_2+\frac{1}{m}\sum_{i=1}^{m}\max\big[1-y_i(\langle\mathbf{w},\mathbf{x}_i\rangle+b),0\big].$$

Let $\gamma_{m,c} = 1 - M_{m,c}/m$. The proof follows since $M_{m,c}/m \to 1$ almost surely for any $c > 0$. Notice by Inequality (14) we have

$$\mathbb{P}\Big(\gamma_{m,c} \geq \lambda\Big) \leq \exp\Big(-m\lambda^2/8 + (T_c+1)\log 2\Big), \tag{15}$$

that is, the convergence is uniform in $\mathbb{P}$.

We have shown that the average testing error is upper bounded. The final step is to show that this implies that in fact the random variable given by the conditional expectation (conditioned on the training sample) of the error is bounded almost surely as in the statement of the theorem. To make things precise, consider a fixed m, and let $\omega_1 \in \Omega_1$ and $\omega_2 \in \Omega_2$ generate the m training samples and m testing samples, respectively, and for shorthand let $\mathcal{T}^m$ denote the random variable of the first m training samples. Let us denote the probability measures for the training by ρ_1 and the testing samples by ρ_2. By independence, the joint measure is given by the product of these two. We rely on this property in what follows. Now fix a λ and a $c > 0$. In our new notation, Equation (15) now reads:

$$\begin{aligned}\int_{\Omega_1}\int_{\Omega_2} \mathbf{1}\big\{\gamma_{m,c}(\omega_1,\omega_2) \geq \lambda\big\}\, d\rho_2(\omega_2)\, d\rho_1(\omega_1) &= \mathbb{P}\Big(\gamma_{m,c}(\omega_1,\omega_2) \geq \lambda\Big) \\ &\leq \exp\Big(-m\lambda^2/8 + (T_c+1)\log 2\Big).\end{aligned}$$

We now bound $\mathbb{P}_{\omega_1}(\mathbb{E}_{\omega_2}[\gamma_{m,c}(\omega_1,\omega_2)\,|\,\mathcal{T}^m] > \lambda)$, and then use Borel-Cantelli to show that this event can happen only finitely often. We have:

$$\begin{aligned}
& \mathbb{P}_{\omega_1}(\mathbb{E}_{\omega_2}[\gamma_{m,c}(\omega_1,\omega_2)\,|\,\mathcal{T}^m] > \lambda) \\
= & \int_{\Omega_1} \mathbf{1}\Big\{ \int_{\Omega_2} \gamma_{m,c}(\omega_1,\omega_2)\, d\rho_2(\omega_2) > \lambda\Big\}\, d\rho_1(\omega_1) \\
\leq & \int_{\Omega_1} \mathbf{1}\Big\{ \Big[\int_{\Omega_2} \gamma_{m,c}(\omega_1,\omega_2)\mathbf{1}(\gamma_{m,c}(\omega_1,\omega_2) \leq \lambda)\, d\rho_2(\omega_2) + \\
& \int_{\Omega_2} \gamma_{m,c}(\omega_1,\omega_2)\mathbf{1}(\gamma_{m,c}(\omega_1,\omega_2) > \lambda)\, d\rho_2(\omega_2)\Big] \geq 2\lambda\Big\} d\rho_1(\omega_1) \\
\leq & \int_{\Omega_1} \mathbf{1}\Big\{ \Big[\int_{\Omega_2} \lambda\mathbf{1}(\lambda(\omega_1,\omega_2) \leq \lambda)\, d\rho_2(\omega_2) + \\
& \int_{\Omega_2} \mathbf{1}(\gamma_{m,c}(\omega_1,\omega_2) > \lambda)\, d\rho_2(\omega_2)\Big] \geq 2\lambda\Big\} d\rho_1(\omega_1) \\
\leq & \int_{\Omega_1} \mathbf{1}\Big\{ \Big[\lambda + \int_{\Omega_2} \mathbf{1}(\gamma_{m,c}(\omega_1,\omega_2) > \lambda)\, d\rho_2(\omega_2)\Big] \geq 2\lambda\Big\} d\rho_1(\omega_1) \\
= & \int_{\Omega_1} \mathbf{1}\Big\{ \int_{\Omega_2} \mathbf{1}(\gamma_{m,c}(\omega_1,\omega_2) > \lambda)\, d\rho_2(\omega_2) \geq \lambda\Big\} d\rho_1(\omega_1).
\end{aligned}$$

Here, the first equality holds because training and testing samples are independent, and hence the joint measure is the product of ρ_1 and ρ_2. The second inequality holds because $\gamma_{m,c}(\omega_1,\omega_2) \leq 1$ everywhere. Further notice that

$$\begin{aligned}
& \int_{\Omega_1}\int_{\Omega_2} \mathbf{1}\big\{\gamma_{m,c}(\omega_1,\omega_2) \geq \lambda\big\}\, d\rho_2(\omega_2)\, d\rho_1(\omega_1) \\
& \geq \int_{\Omega_1} \lambda\mathbf{1}\Big\{ \int_{\Omega_2} \mathbf{1}\big(\gamma_{m,c}(\omega_1,\omega_2) \geq \lambda\big)\, d\rho(\omega_2) > \lambda\Big\}\, d\rho_1(\omega_1).
\end{aligned}$$

Thus we have

$$\mathbb{P}(\mathbb{E}_{\omega_2}(\gamma_{m,c}(\omega_1,\omega_2))>\lambda)\leq \mathbb{P}\Big(\gamma_{m,c}(\omega_1,\omega_2)\geq\lambda\Big)/\lambda\leq \exp\Big(-m\lambda^2/8+(T_c+1)\log 2\Big)/\lambda.$$

For any λ and c, summing up the right hand side over $m=1$ to ∞ is finite, hence the theorem follows from the Borel-Cantelli lemma. ■

Remark 10 We note that M_m/m converges to 1 almost surely even if $\mathcal{X}$ is not bounded. Indeed, to see this, fix $\varepsilon>0$, and let $\mathcal{X}'\subseteq\mathcal{X}$ be a bounded set such that $\mathbb{P}(\mathcal{X}')>1-\varepsilon$. Then, with probability one,

$$\#(\text{unpaired samples in}\mathcal{X}')/m\to 0,$$

by Lemma 9. In addition,

$$\max\big(\#(\text{training samples not in }\mathcal{X}'),\#(\text{testing samples not in }\mathcal{X}')\big)/m\to\varepsilon.$$

Notice that

$$\begin{aligned}M_m\geq m&-\#(\text{unpaired samples in }\mathcal{X}')\\&-\max\big(\#(\text{training samples not in }\mathcal{X}'),\#(\text{testing samples not in }\mathcal{X}')\big).\end{aligned}$$

Hence

$$\lim_{m\to\infty}M_m/m\geq 1-\varepsilon,$$

almost surely. Since ε is arbitrary, we have $M_m/m\to 1$ almost surely.

Next, we prove an analog of Theorem 8 for the kernelized case, and then show that these two imply statistical consistency of linear and kernelized SVMs. Again, let $\mathcal{X}\subseteq\mathbb{R}^n$ be bounded, and suppose the training samples $(\mathbf{x}_i,y_i)_{i=1}^\infty$ are generated i.i.d. according to an unknown distribution $\mathbb{P}$ supported on $\mathcal{X}\times\{-1,+1\}$.

Theorem 11 *Denote* $K\triangleq\max_{\mathbf{x}\in\mathcal{X}}k(\mathbf{x},\mathbf{x})$. *Suppose there exists* $\rho>0$ *and a continuous non-decreasing function* $f:\mathbb{R}^+\to\mathbb{R}^+$ *satisfying* $f(0)=0$, *such that:*

$$k(\mathbf{x},\mathbf{x})+k(\mathbf{x}',\mathbf{x}')-2k(\mathbf{x},\mathbf{x}')\leq f(\|\mathbf{x}-\mathbf{x}'\|_2^2),\quad\forall\mathbf{x},\mathbf{x}'\in\mathcal{X},\|\mathbf{x}-\mathbf{x}'\|_2\leq\rho.$$

Then there exists a random sequence $\{\gamma_{m,c}\}$ *such that:*

1. $\forall c>0$, $\lim_{m\to\infty}\gamma_{m,c}=0$ *almost surely, and the convergence is uniform in* $\mathbb{P}$*;*
2. *the following bounds on the Bayes loss and the hinge loss hold uniformly for all* $(\mathbf{w},b)\in\mathcal{H}\times\mathbb{R}$

$$\begin{aligned}&\mathbb{E}_{\mathbb{P}}(\mathbf{1}_{y\neq sgn(\langle\mathbf{w},\Phi(\mathbf{x})\rangle+b)})\leq\gamma_{m,c}+c\|\mathbf{w}\|_{\mathcal{H}}+\frac{1}{m}\sum_{i=1}^m\max\big[1-y_i(\langle\mathbf{w},\Phi(\mathbf{x}_i)\rangle+b),0\big],\\&\mathbb{E}_{(\mathbf{x},y)\sim\mathbb{P}}\big(\max(1-y(\langle\mathbf{w},\Phi(\mathbf{x})\rangle+b),0)\big)\leq\\&\qquad\gamma_{m,c}(1+K\|\mathbf{w}\|_{\mathcal{H}}+|b|)+c\|\mathbf{w}\|_{\mathcal{H}}+\frac{1}{m}\sum_{i=1}^m\max\big[1-y_i(\langle\mathbf{w},\Phi(\mathbf{x}_i)\rangle+b),0\big].\end{aligned}$$

Proof As in the proof of Theorem 8, we generate a set of m testing samples and m training samples, and then lower-bound the number of samples that can form a *sample pair* in the feature-space; that is, a pair consisting of a training sample $(\mathbf{x}, y)$ and a testing sample $(\mathbf{x}', y')$ such that $y = y'$ and $\|\Phi(\mathbf{x}) - \Phi(\mathbf{x}')\|_{\mathcal{H}} \leq c$. In contrast to the finite-dimensional sample space, the feature space may be infinite dimensional, and thus our decomposition may have an infinite number of "bricks." In this case, our multinomial random variable argument used in the proof of Lemma 9 breaks down. Nevertheless, we are able to lower bound the number of sample pairs in the feature space by the number of sample pairs in the *sample space*.

Define $f^{-1}(\alpha) \triangleq \max\{\beta \geq 0 | f(\beta) \leq \alpha\}$. Since $f(\cdot)$ is continuous, $f^{-1}(\alpha) > 0$ for any $\alpha > 0$. Now notice that by Lemma 7, if a testing sample $\mathbf{x}$ and a training sample $\mathbf{x}'$ belong to a "brick" with length of each side $\min(\rho/\sqrt{n}, f^{-1}(c^2)/\sqrt{n})$ in the *sample space* (see the proof of Lemma 9), $\|\Phi(\mathbf{x}) - \Phi(\mathbf{x}')\|_{\mathcal{H}} \leq c$. Hence the number of *sample pairs* in the feature space is lower bounded by the number of pairs of samples that fall in the same brick in the sample space. We can cover $\mathcal{X}$ with finitely many (denoted as T_c) such bricks since $f^{-1}(c^2) > 0$. Then, a similar argument as in Lemma 9 shows that the ratio of samples that form pairs in a brick converges to 1 as m increases. Further notice that for M paired samples, the total testing error and hinge-loss are both upper-bounded by

$$cM\|\mathbf{w}\|_{\mathcal{H}} + \sum_{i=1}^{M} \max\left[1 - y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b), 0\right].$$

The rest of the proof is identical to Theorem 8. In particular, Inequality (15) still holds. ■

Note that the condition in Theorem 11 is satisfied by most commonly used kernels, for example, homogeneous polynominal kernels and Gaussian radial basis functions. This condition requires that the feature mapping is "smooth" and hence preserves "locality" of the disturbance, that is, small disturbance in the sample space guarantees the corresponding disturbance in the feature space is also small. It is easy to construct non-smooth kernel functions which do not generalize well. For example, consider the following kernel:

$$k(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 & \mathbf{x} = \mathbf{x}'; \\ 0 & \mathbf{x} \neq \mathbf{x}'. \end{cases}$$

A standard RKHS regularized SVM using this kernel leads to a decision function

$$\text{sign}(\sum_{i=1}^{m} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b),$$

which equals $\text{sign}(b)$ and provides no meaningful prediction if the testing sample $\mathbf{x}$ is not one of the training samples. Hence as m increases, the testing error remains as large as 50% regardless of the tradeoff parameter used in the algorithm, while the training error can be made arbitrarily small by fine-tuning the parameter.

5.1 Convergence to Bayes Risk

Next we relate the results of Theorem 8 and Theorem 11 to the standard consistency notion, that is, convergence to the Bayes Risk (Steinwart, 2005). The key point of interest in our proof is the use of a robustness condition in place of a VC-dimension or stability condition used in Steinwart (2005). The proof in Steinwart (2005) has 4 main steps. They show: (i) there always exists a minimizer to

the expected regularized (kernel) hinge loss; (ii) the expected regularized hinge loss of the minimizer converges to the expected hinge loss as the regularizer goes to zero; (iii) if a sequence of functions asymptotically have optimal expected hinge loss, then they also have optimal expected loss; and (iv) the expected hinge loss of the minimizer of the regularized *training* hinge loss concentrates around the empirical regularized hinge loss. In Steinwart (2005), this final step, (iv), is accomplished using concentration inequalities derived from VC-dimension considerations, and stability considerations.

Instead, we use our robustness-based results of Theorem 8 and Theorem 11 to replace these approaches (Lemmas 3.21 and 3.22 in Steinwart 2005) in proving step (iv), and thus to establish the main result.

Recall that a classifier is a rule that assigns to every training set $T = \{\mathbf{x}_i, y_i\}_{i=1}^m$ a measurable function f_T. The risk of a measurable function $f : \mathcal{X} \to \mathbb{R}$ is defined as

$$\mathcal{R}_{\mathbb{P}}(f) \triangleq \mathbb{P}(\{\mathbf{x}, y : \text{sign} f(\mathbf{x}) \neq y\}).$$

The smallest achievable risk

$$\mathcal{R}_{\mathbb{P}} \triangleq \inf\{\mathcal{R}_{\mathbb{P}}(f) | f : \mathcal{X} \to \mathbb{R} \,\text{measurable}\}$$

is called the *Bayes Risk* of $\mathbb{P}$. A classifier is said to be strongly uniformly consistent if for all distributions P on $\mathcal{X} \times [-1, +1]$, the following holds almost surely.

$$\lim_{m\to\infty} \mathcal{R}_{\mathbb{P}}(f_T) = \mathcal{R}_{\mathbb{P}}.$$

Without loss of generality, we only consider the kernel version. Recall a definition from Steinwart (2005).

Definition 12 *Let $C(\mathcal{X})$ be the set of all continuous functions defined on a compact metric space $\mathcal{X}$. Consider the mapping $I : \mathcal{H} \to C(\mathcal{X})$ defined by $I\mathbf{w} \triangleq \langle \mathbf{w}, \Phi(\cdot)\rangle$. If I has a dense image, we call the kernel* universal.

Roughly speaking, if a kernel is universal, then the corresponding RKHS is rich enough to satisfy the condition of step (ii) above.

Theorem 13 *If a kernel satisfies the condition of Theorem 11, and is universal, then the Kernel SVM with $c \downarrow 0$ sufficiently slowly is strongly uniformly consistent.*

Proof We first introduce some notation, largely following Steinwart (2005). For some probability measure μ and $(\mathbf{w}, b) \in \mathcal{H} \times \mathbb{R}$,

$$R_{L,\mu}((\mathbf{w}, b)) \triangleq \mathbb{E}_{(\mathbf{x},y)\sim\mu}\big\{\max(0, 1 - y(\langle \mathbf{w}, \Phi(\mathbf{x})\rangle + b))\big\},$$

is the expected hinge-loss under probability μ, and

$$R^c_{L,\mu}((\mathbf{w}, b)) \triangleq c\|\mathbf{w}\|_{\mathcal{H}} + \mathbb{E}_{(\mathbf{x},y)\sim\mu}\big\{\max(0, 1 - y(\langle \mathbf{w}, \Phi(\mathbf{x})\rangle + b))\big\}$$

is the regularized expected hinge-loss. Hence $R_{L,\mathbb{P}}(\cdot)$ and $R^c_{L,\mathbb{P}}(\cdot)$ are the expected hinge-loss and regularized expected hinge-loss under the generating probability $\mathbb{P}$. If μ is the empirical distribution of m samples, we write $R_{L,m}(\cdot)$ and $R^c_{L,m}(\cdot)$ respectively. Notice $R^c_{L,m}(\cdot)$ is the objective function of the SVM. Denote its solution by $f_{m,c}$, that is, the classifier we get by running SVM with m samples

and parameter c. Further denote by $f_{\mathbb{P},c} \in \mathcal{H} \times \mathbb{R}$ the minimizer of $R^c_{L,\mathbb{P}}(\cdot)$. The existence of such a minimizer is proved in Lemma 3.1 of Steinwart (2005) (step (i)). Let

$$\mathcal{R}_{L,\mathbb{P}} \triangleq \min_{f \text{ measurable}} \mathbb{E}_{\mathbf{x},y\sim\mathbb{P}}\Big\{ \max\big(1 - yf(\mathbf{x}), 0\big)\Big\},$$

that is, the smallest achievable hinge-loss for all measurable functions.

The main content of our proof is to use Theorems 8 and 11 to prove step (iv) in Steinwart (2005). In particular, we show: if $c \downarrow 0$ "slowly", we have with probability one

$$\lim_{m\to\infty} R_{L,\mathbb{P}}(f_{m,c}) = \mathcal{R}_{L,\mathbb{P}}. \tag{16}$$

To prove Equation (16), denote by $\mathbf{w}(f)$ and $b(f)$ as the weight part and offset part of any classifier f. Next, we bound the magnitude of $f_{m,c}$ by using $R^c_{L,m}(f_{m,c}) \le R^c_{L,m}(\mathbf{0},0) \le 1$, which leads to

$$\|\mathbf{w}(f_{m,c})\|_{\mathcal{H}} \le 1/c$$

and

$$|b(f_{m,c})| \le 2 + K\|\mathbf{w}(f_{m,c})\|_{\mathcal{H}} \le 2 + K/c.$$

From Theorem 11 (note that the bound holds uniformly for all $(\mathbf{w}, b)$), we have

$$\begin{aligned}
R_{L,\mathbb{P}}(f_{m,c}) &\le \gamma_{m,c}[1 + K\|\mathbf{w}(f_{m,c})\|_{\mathcal{H}} + |b|] + R^c_{L,m}(f_{m,c}) \\
&\le \gamma_{m,c}[3 + 2K/c] + R^c_{L,m}(f_{m,c}) \\
&\le \gamma_{m,c}[3 + 2K/c] + R^c_{L,m}(f_{\mathbb{P},c}) \\
&= \mathcal{R}_{L,\mathbb{P}} + \gamma_{m,c}[3 + 2K/c] + \big\{R^c_{L,m}(f_{\mathbb{P},c}) - R^c_{L,\mathbb{P}}(f_{\mathbb{P},c})\big\} + \big\{R^c_{L,\mathbb{P}}(f_{\mathbb{P},c}) - \mathcal{R}_{L,\mathbb{P}}\big\} \\
&= \mathcal{R}_{L,\mathbb{P}} + \gamma_{m,c}[3 + 2K/c] + \big\{R_{L,m}(f_{\mathbb{P},c}) - R_{L,\mathbb{P}}(f_{\mathbb{P},c})\big\} + \big\{R^c_{L,\mathbb{P}}(f_{\mathbb{P},c}) - \mathcal{R}_{L,\mathbb{P}}\big\}.
\end{aligned}$$

The last inequality holds because $f_{m,c}$ minimizes $R^c_{L,m}$.

It is known (Steinwart, 2005, Proposition 3.2) (step (ii)) that if the kernel used is rich enough, that is, universal, then

$$\lim_{c\to 0} R^c_{L,\mathbb{P}}(f_{\mathbb{P}}, c) = \mathcal{R}_{L,\mathbb{P}}.$$

For fixed $c > 0$, we have

$$\lim_{m\to\infty} R_{L,m}(f_{\mathbb{P},c}) = R_{L,\mathbb{P}}(f_{\mathbb{P},c}),$$

almost surely due to the strong law of large numbers (notice that $f_{\mathbb{P},c}$ is a fixed classifier), and $\gamma_{m,c}[3 + 2K/c] \to 0$ almost surely. Notice that neither convergence rate depends on $\mathbb{P}$. Therefore, if $c \downarrow 0$ sufficiently slowly,[3] we have almost surely

$$\lim_{m\to\infty} R_{L,\mathbb{P}}(f_{m,c}) \le \mathcal{R}_{L,\mathbb{P}}.$$

Now, for any m and c, we have $R_{L,\mathbb{P}}(f_{m,c}) \ge \mathcal{R}_{L,\mathbb{P}}$ by definition. This implies that Equation (16) holds almost surely, thus giving us step (iv).

Finally, Proposition 3.3. of Steinwart (2005) shows step (iii), namely, approximating hinge loss is sufficient to guarantee approximation of the Bayes loss. Thus Equation (16) implies that the risk

3. For example, we can take $\{c(m)\}$ be the smallest number satisfying $c(m) \ge m^{-1/8}$ and $T_{c(m)} \le m^{1/8}/\log 2 - 1$. Inequality (15) thus leads to $\sum_{m=1}^{\infty} P(\gamma_{m,c(m)}/c(m) \ge m^{1/4}) \le +\infty$ which implies uniform convergence of $\gamma_{m,c(m)}/c(m)$.

of function $f_{m,c}$ converges to Bayes risk. ■

Before concluding this section, we remark that although we focus in this paper the hinge-loss function and the RKHS norm regularizer, the robustness approach to establish consistency can be generalized to other regularization schemes and loss functions. Indeed, throughout the proof we only require that the regularized loss (that is, the training loss plus the regularization penalty) is an upper bound of the minimax error with respect to certain set-inclusive uncertainty. This is a property satisfied by many classification algorithms even though an exact equivalence relationship similar to the one presented in this paper may not exist. This suggests using the robustness view to derive sharp sample complexity bounds for a broad class of algorithms (e.g., Steinwart and Christmann, 2008).

6. Concluding Remarks

This work considers the relationship between robust and regularized SVM classification. In particular, we prove that the standard norm-regularized SVM classifier is in fact the solution to a robust classification setup, and thus known results about regularized classifiers extend to robust classifiers. To the best of our knowledge, this is the first explicit such link between regularization and robustness in pattern classification. The interpretation of this link is that norm-based regularization essentially builds in a robustness to sample noise whose probability level sets are symmetric unit balls with respect to the dual of the regularizing norm. It would be interesting to understand the performance gains possible when the noise does not have such characteristics, and the robust setup is used in place of regularization with appropriately defined uncertainty set.

Based on the robustness interpretation of the regularization term, we re-proved the consistency of SVMs without direct appeal to notions of metric entropy, VC-dimension, or stability. Our proof suggests that the ability to handle disturbance is crucial for an algorithm to achieve good generalization ability. In particular, for "smooth" feature mappings, the robustness to disturbance in the observation space is guaranteed and hence SVMs achieve consistency. On the other-hand, certain "non-smooth" feature mappings fail to be consistent simply because for such kernels the robustness in the feature-space (guaranteed by the regularization process) does not imply robustness in the observation space.

Acknowledgments

We thank the editor and three anonymous reviewers for significantly improving the accessibility of this manuscript. We also benefited from comments from participants in ITA 2008 and at a NIPS 2008 workshop on optimization. This research was partially supported by the Canada Research Chairs Program, by the Israel Science Foundation (contract 890015), by a Horev Fellowship, by NSF Grants EFRI-0735905, CNS-0721532, and a grant from DTRA.

Appendix A.

In this appendix we show that for RBF kernels, it is possible to relate robustness in the feature space and robustness in the sample space more directly.

Theorem 14 *Suppose the Kernel function has the form $k(\mathbf{x},\mathbf{x}') = f(\|\mathbf{x}-\mathbf{x}'\|)$, with $f:\mathbb{R}^+\to\mathbb{R}$ a decreasing function. Denote by $\mathcal{H}$ the RKHS space of $k(\cdot,\cdot)$ and $\Phi(\cdot)$ the corresponding feature mapping. Then we have for any $\mathbf{x}\in\mathbb{R}^n$, $\mathbf{w}\in\mathcal{H}$ and $c>0$,*

$$\sup_{\|\delta\|\le c}\langle \mathbf{w}, \Phi(\mathbf{x}-\delta)\rangle = \sup_{\|\delta_\phi\|_{\mathcal{H}}\le\sqrt{2f(0)-2f(c)}}\langle \mathbf{w}, \Phi(\mathbf{x})+\delta_\phi\rangle.$$

Proof We show that the left-hand-side is not larger than the right-hand-side, and vice versa.

First we show

$$\sup_{\|\delta\|\le c}\langle \mathbf{w}, \Phi(\mathbf{x}-\delta)\rangle \le \sup_{\|\delta_\phi\|_{\mathcal{H}}\le\sqrt{2f(0)-2f(c)}}\langle \mathbf{w}, \Phi(\mathbf{x})-\delta_\phi\rangle. \tag{17}$$

We notice that for any $\|\delta\|\le c$, we have

$$\begin{aligned}
&\langle \mathbf{w}, \Phi(\mathbf{x}-\delta)\rangle\\
=&\Big\langle \mathbf{w}, \Phi(\mathbf{x})+\big(\Phi(\mathbf{x}-\delta)-\Phi(\mathbf{x})\big)\Big\rangle\\
=&\langle \mathbf{w}, \Phi(\mathbf{x})\rangle + \langle \mathbf{w}, \Phi(\mathbf{x}-\delta)-\Phi(\mathbf{x})\rangle\\
\le&\langle \mathbf{w}, \Phi(\mathbf{x})\rangle + \|\mathbf{w}\|_{\mathcal{H}}\cdot\|\Phi(\mathbf{x}-\delta)-\Phi(\mathbf{x})\|_{\mathcal{H}}\\
\le&\langle \mathbf{w}, \Phi(\mathbf{x})\rangle + \|\mathbf{w}\|_{\mathcal{H}}\sqrt{2f(0)-2f(c)}\\
=&\sup_{\|\delta_\phi\|_{\mathcal{H}}\le\sqrt{2f(0)-2f(c)}}\langle \mathbf{w}, \Phi(\mathbf{x})-\delta_\phi\rangle.
\end{aligned}$$

Taking the supremum over δ establishes Inequality (17).

Next, we show the opposite inequality,

$$\sup_{\|\delta\|\le c}\langle \mathbf{w}, \Phi(\mathbf{x}-\delta)\rangle \ge \sup_{\|\delta_\phi\|_{\mathcal{H}}\le\sqrt{2f(0)-2f(c)}}\langle \mathbf{w}, \Phi(\mathbf{x})-\delta_\phi\rangle. \tag{18}$$

If $f(c)=f(0)$, then Inequality 18 holds trivially, hence we only consider the case that $f(c)<f(0)$. Notice that the inner product is a continuous function in $\mathcal{H}$, hence for any $\varepsilon>0$, there exists a δ'_ϕ such that

$$\langle \mathbf{w}, \Phi(\mathbf{x})-\delta'_\phi\rangle > \sup_{\|\delta_\phi\|_{\mathcal{H}}\le\sqrt{2f(0)-2f(c)}}\langle \mathbf{w}, \Phi(\mathbf{x})-\delta_\phi\rangle-\varepsilon;\quad \|\delta'_\phi\|_{\mathcal{H}}<\sqrt{2f(0)-2f(c)}.$$

Recall that the RKHS space is the completion of the feature mapping, thus there exists a sequence of $\{\mathbf{x}'_i\}\in\mathbb{R}^n$ such that

$$\Phi(\mathbf{x}'_i)\to\Phi(\mathbf{x})-\delta'_\phi, \tag{19}$$

which is equivalent to

$$\big(\Phi(\mathbf{x}'_i)-\Phi(\mathbf{x})\big)\to-\delta'_\phi.$$

This leads to

$$\begin{aligned}
&\lim_{i\to\infty}\sqrt{2f(0)-2f(\|\mathbf{x}'_i-\mathbf{x}\|)}\\
=&\lim_{i\to\infty}\|\Phi(\mathbf{x}'_i)-\Phi(\mathbf{x})\|_{\mathcal{H}}\\
=&\|\delta'_\phi\|_{\mathcal{H}}<\sqrt{2f(0)-2f(c)}.
\end{aligned}$$

Since f is decreasing, we conclude that $\|\mathbf{x}_i' - \mathbf{x}\| \leq c$ holds except for a finite number of i. By (19) we have

$$\langle \mathbf{w}, \Phi(\mathbf{x}_i') \rangle \to \langle \mathbf{w}, \Phi(\mathbf{x}) - \delta_\phi' \rangle > \sup_{\|\delta_\phi\|_{\mathcal{H}} \leq \sqrt{2f(0)-2f(c)}} \langle \mathbf{w}, \Phi(\mathbf{x}) - \delta_\phi \rangle - \varepsilon,$$

which means

$$\sup_{\|\delta\| \leq c} \langle \mathbf{w}, \Phi(\mathbf{x} - \delta) \rangle \geq \sup_{\|\delta_\phi\|_{\mathcal{H}} \leq \sqrt{2f(0)-2f(c)}} \langle \mathbf{w}, \Phi(\mathbf{x}) - \delta_\phi \rangle - \varepsilon.$$

Since ε is arbitrary, we establish Inequality (18).

Combining Inequality (17) and Inequality (18) proves the theorem. ■

References

M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *The Journal of Machine Learning Research*, 3:463–482, November 2002.

P. L. Bartlett, O. Bousquet, and S. Mendelson. Local Rademacher complexities. *The Annals of Statistics*, 33(4):1497–1537, 2005.

A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, August 1999.

K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1(1):23–34, 1992.

D. Bertsimas and A. Fertis. Personal Correspondence, March 2008.

D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, January 2004.

C. Bhattacharyya. Robust classification of noisy data using second order cone programming approach. In *Proceedings International Conference on Intelligent Sensing and Information Processing*, pages 433–438, Chennai, India, 2004.

C. Bhattacharyya, L. R. Grate, M. I. Jordan, L. El Ghaoui, and I. S. Mian. Robust sparse hyperplane classifiers: Application to uncertain molecular profiling data. *Journal of Computational Biology*, 11(6):1073–1089, 2004a.

C. Bhattacharyya, K. S. Pannagadatta, and A. J. Smola. A second order cone programming formulation for classifying missing data. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems (NIPS17)*, Cambridge, MA, 2004b. MIT Press.

J. Bi and T. Zhang. Support vector classification with input data uncertainty. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems (NIPS17)*, Cambridge, MA, 2004. MIT Press.

C. M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995. doi: 10.1162/neco.1995.7.1.108. URL http://www.mitpressjournals.org/doi/abs/10.1162/neco.1995.7.1.108.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152, New York, NY, 1992.

O. Bousquet and A. Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

A. Christmann and I. Steinwart. On robustness properties of convex risk minimization methods for pattern recognition. *The Journal of Machine Learning Research*, 5:1007–1034, 2004.

A. Christmann and I. Steinwart. Consistency and robustness of kernel based regression in convex risk minimization. *Bernoulli*, 13(3):799–819, 2007.

A. Christmann and A. Van Messem. Bouligand derivatives and robustness of support vector machines for regression. *The Journal of Machine Learning Research*, 9:915–936, 2008.

C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

R. Durrett. *Probability: Theory and Examples*. Duxbury Press, 2004.

L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18:1035–1064, 1997.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 171–203, Cambridge, MA, 2000. MIT Press.

A. Globerson and S. Roweis. Nightmare at test time: Robust learning by feature deletion. In *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 353–360, New York, NY, USA, 2006. ACM Press.

F. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974.

F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley & Sons, New York, 1986.

P. J. Huber. *Robust Statistics*. John Wiley & Sons, New York, 1981.

M. Kearns, Y. Mansour, A. Ng, and D. Ron. An experimental and theoretical comparison of model selection methods. *Machine Learning*, 27:7–50, 1997.

V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1):1–50, 2002.

S. Kutin and P. Niyogi. Almost-everywhere algorithmic stability and generalization error. In *UAI-2002: Uncertainty in Artificial Intelligence*, pages 275–282, 2002.

G. R. Lanckriet, L. El Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *The Journal of Machine Learning Research*, 3:555–582, 2003.

R. A. Maronna, R. D. Martin, and V. J. Yohai. *Robust Statistics: Theory and Methods*. John Wiley & Sons, New York, 2006.

S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin. Learning theory: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, 25(1-3):161–193, 2006.

T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. General conditions for predictivity in learning theory. *Nature*, 428(6981):419–422, 2004.

P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, New York, 1987.

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

P. K. Shivaswamy, C. Bhattacharyya, and A. J. Smola. Second order cone programming approaches for handling missing and uncertain data. *The Journal of Machine Learning Research*, 7:1283–1314, July 2006.

A. J. Smola, B. Schölkopf, and K. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998.

I. Steinwart. Consistency of support vector machines and other regularized kernel classifiers. *IEEE Transactions on Information Theory*, 51(1):128–142, 2005.

I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, New York, 2008.

C. H. Teo, A. Globerson, S. Roweis, and A. J. Smola. Convex learning with invariances. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1489–1496, Cambridge, MA, 2008. MIT Press.

T. Trafalis and R. Gilbert. Robust support vector machines for classification and computational issues. *Optimization Methods and Software*, 22(1):187–198, February 2007.

A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes*. Springer-Verlag, New York, 2000.

V. N. Vapnik and A. Chervonenkis. *Theory of Pattern Recognition*. Nauka, Moscow, 1974.

V. N. Vapnik and A. Chervonenkis. The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, 1(3):260–284, 1991.

V. N. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:744–780, 1963.

H. Xu, C. Caramanis, and S. Mannor. Robust regression and Lasso. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1801–1808, 2009.

Journal of Machine Learning Research 10 (2009) 1511-1526 Submitted 2/08; Revised 3/09; Published 7/09

Strong Limit Theorems for the Bayesian Scoring Criterion in Bayesian Networks

Nikolai Slobodianik NIKOLAI.SLOBODIANIK@GMAIL.COM
Department of Mathematics and Statistics
York University
4700 Keele Street, Toronto, ON M3J 1P3, Canada

Dmitry Zaporozhets ZAP1979@GMAIL.COM
Steklov Institute of Mathematics at St. Petersburg
27 Fontanka, St. Petersburg 191023, Russia

Neal Madras MADRAS@MATHSTAT.YORKU.CA
Department of Mathematics and Statistics
York University
4700 Keele Street, Toronto, ON M3J 1P3, Canada

Editor: Max Chickering

Abstract

In the machine learning community, the Bayesian scoring criterion is widely used for model selection problems. One of the fundamental theoretical properties justifying the usage of the Bayesian scoring criterion is its consistency. In this paper we refine this property for the case of binomial Bayesian network models. As a by-product of our derivations we establish strong consistency and obtain the law of iterated logarithm for the Bayesian scoring criterion.

Keywords: Bayesian networks, consistency, scoring criterion, model selection, BIC

1. Introduction

Bayesian networks are graphical structures which characterize probabilistic relationships among variables of interest and serve as a ground model for doing probabilistic inference in large systems of interdependent components. A basic element of a Bayesian network is a directed acyclic graph (DAG) which is bound to an underlying joint probability distribution by the Markov condition. The absence of certain arcs (edges) in a DAG encodes conditional independences in this distribution. DAG's not only provide a starting point for implementation of inference and parameter learning algorithms, but they also, due to their graphical nature, offer an intuitive picture of the relationships among the variables. It happens too often that researchers have only a random sample from a probability distribution and face a problem of choosing the appropriate DAG between a large number of competing structures. This, effectively, constitutes the model selection problem in the space of Bayesian networks. The methodology which is concerned with solving such task is called Bayesian structure learning.

Suppose that the data consists of n i.i.d. random vectors $X_1, \ldots, X_n$ with each X_i having the unknown probability distribution P. We define a probability space Ω with measure Pr for infinite i.i.d. sequences $X_1, X_2, \ldots$ having distribution P. There are many structures which can form a Bayesian

network with the distribution P (see Section 2 for formal definitions and examples), however not all of them are optimal for future analysis. Indeed, since the presence of an arc (edge) does not necessarily guarantee direct dependency between corresponding variables, a complete DAG constitutes a Bayesian network with any probability distribution, yet provides no information about conditional independences in P. It is natural to seek structures which not only form a Bayesian network with P, but also entail only conditional independences in this distribution. These DAGs are called *faithful* to P or else *perfect maps* of P. Unfortunately, it turns out that not all probability distributions have an associated faithful structure. In this case it is desirable to identify a structure which satisfies certain "optimality" properties with respect to P. Roughly speaking, we want to include only those edges that are necessary for describing P.

A *scoring criterion* for DAGs is a function that assigns a value to each DAG under consideration based on the data. Suppose M is the set of all DAGs of a fixed size. Under the Bayesian approach to structure learning, the DAG m is chosen from M such that m maximizes the posterior probability given the observed data D:

$$p(m|D,\psi) = \frac{p(m|\psi)p(D|m,\psi)}{p(D|\psi)} = \frac{p(m|\psi)\int_{\Omega_m} p(D|m,\Theta_m,\psi)p(\Theta_m|m,\psi)d\Theta_m}{\sum_{m\in M} p(m|\psi)\int_{\Omega_m} p(D|m,\Theta_m,\psi)p(\Theta_m|m,\psi)d\Theta_m}, \quad (1)$$

where Θ_m denotes the set of parameters of the conditional distributions of each "node given its parents" for all the nodes of the DAG m, Ω_m denotes the domain of these parameters, and ψ denotes the system of parameter priors. The quantity $P(D|m,\psi)$ is called the *marginal likelihood*, *Bayesian scoring criterion* or else *Score* of the graph m. We denote it as $\texttt{score}_B(D|m)$. Assuming $\sum_{m\in M} p(m|\psi) = 1$ for all $m \in M$, the Bayesian scoring criterion provides a measure of posterior certainty of the graph m under the prior system ψ.

It is quite interesting to see if the Bayesian scoring criterion is *consistent*, that is, as the size of data D approaches infinity, the criterion is maximized at the DAG which forms a Bayesian network with P and has smallest dimension. Based on the fundamental results of Haughton (1988) and Geiger et al. (2001), the consistency of Bayesian scoring criterion has been established for the class of multinomial Bayesian networks. Chickering (2002) provides a detailed sketch of the proof. Further, for the same model class, if P admits a faithful DAG representation m, then m has the smallest dimension among all DAGs which form a Bayesian network with P (see, for example, Neapolitan, 2004, Corollary 8.1) . Therefore, due to consistency of the Bayesian scoring criterion, we can conclude that if P admits a faithful DAG representation m then, in the limit, the Bayesian scoring criterion will be maximized at m. This last result is naturally expected: as more information becomes available, a scoring criterion should recognize the properties of the underlying distribution P with increasing precision.

Although the consistency property provides insight into the limiting properties of the posterior distribution over the graph space, it is interesting to know at what rate (as a function of sample size) the graph(s) with the smallest dimension become favored by the Bayesian scoring criterion. In this article we address this question for the case of binomial Bayesian network models. We also show that in addition to being consistent for these models, the Bayesian scoring criterion is also *strongly consistent* (see Definition 4). Our proofs are mostly self-contained, relying mainly on well-known limit theorems of classical probability. At one point we require the input of Haughton (1988) and Geiger et al. (2001) mentioned in the preceding paragraph (but note that their results only deal with consistency, not strong consistency).

It may be possible to re-derive our results using the machinery of VC classes (Vapnik, 1998) or empirical process theory (e.g., van der Vaart and Wellner, 1996), but to our knowledge this has not yet been done. However, one point of our paper is to show that the results are amenable to fairly transparent and accessible proofs, and do not require the overhead of these well-developed theoretical frameworks. That being said, we note that our method assumes that the networks have fixed finite size, and other approaches may be better suited to handling the situation in which the network size gets large.

The rest of the paper is organized as follows. Background and notation appear in Section 2, with some illustrative examples. Our results are presented in Section 3. Section 4 contains some discussion. Proofs appear in the Appendix.

2. Background

A directed graph is a pair (V,E), where $V = \{1,\ldots,N\}$ is a finite set whose elements are called nodes (or vertices), and E is a set of ordered pairs of distinct components of V. Elements of E are called edges (or arcs). If $(i_1,i_2) \in E$ we say that there is an edge from i_1 to i_2. Given a set of nodes $\{i_1,i_2,\ldots,i_k\}$ where $k \geq 2$ and $(i_r,i_{r+1}) \in E$ for $1 \leq r \leq k-1$, we call a sequence of edges $((i_1,i_2),\ldots,(i_{k-1},i_k))$ a path from i_1 to i_k . A path from a node to itself is called a directed cycle. Additionally, a directed graph is called a directed acyclic graph (DAG) if it contains no directed cycles. Given a DAG $m = (V,E)$, a node i_2 is called a parent of i_1 if there is an edge from i_2 to i_1. We write $\mathrm{Pa}(i)$ to denote the set of parents of a node i. A node i_2 is called a descendant of i_1 if there is a path from i_1 to i_2, and i_2 is called a nondescendant of i_1 if i_2 is not a descendant of i_1.

Suppose $m = (V,E)$ is a DAG, and $X = \{\xi_1,\ldots,\xi_N\}$ is a random vector that follows a joint probability distribution P. For each i, let ξ_i correspond to the i^{th} node of V. For $A \subset V$, let ξ_A denote the collection of variables $\{\xi_i : i \in A\}$. (In the literature, sometimes this collection is written simply as A. We will occasionally following this convention, but in mathematical expressions about probabilities we usually prefer to distinguish clearly between the set of variables A and their values ξ_A.) In particular, $\xi_{\mathrm{Pa}(i)}$ describes the states of the parents of node i. We say that (m,P) satisfies the Markov condition if each component of X is conditionally independent of the set of all its nondescendants given the set of all its parents. Finally, if (m,P) satisfies the Markov condition, then we say that (m,P) is a Bayesian network, and that m forms a Bayesian network with P. See Neapolitan (2004) for more details.

The independence constraints encoded in a Bayesian network allow for a simplification of the joint probability distribution P which is captured by the factorization theorem (Neapolitan, 2004, Theorem 1.4):

Theorem 1 *If (m,P) satisfies the Markov condition, then P is equal to the product of its conditional distributions of all nodes given the values of their parents, whenever these conditional distributions exist:*

$$P(\xi_1,\ldots,\xi_N) = \prod_{i=1}^{N} P\left(\xi_i|\xi_{Pa(i)}\right).$$

Consider the following example (also see Neapolitan, 2004, Example 2.9). Rewrite the variables $(\xi_1,\xi_2,\xi_3,\xi_4) = (U,Y,Z,W)$. Suppose we have a Bayesian network (m,P) where m is shown in Figure 1 and the distribution P satisfies the conditions presented in Table 1 for some

$P(u_1) = a$	$P(y_1\|u_1) = 1-(b+c)$	$P(z_1\|y_1) = e$	$P(w_1\|z_1) = g$
$P(u_2) = 1-a$	$P(y_2\|u_1) = c$	$P(z_2\|y_1) = 1-e$	$P(w_2\|z_1) = 1-g$
	$P(y_3\|u_1) = b$	$P(z_1\|y_2) = e$	$P(w_1\|z_2) = h$
	$P(y_1\|u_2) = 1-(b+d)$	$P(z_2\|y_2) = 1-e$	$P(w_2\|z_2) = 1-h$
	$P(y_2\|u_2) = d$	$P(z_1\|y_3) = f$	
	$P(y_3\|u_2) = b$	$P(z_2\|y_3) = 1-f$	

Table 1: Constraints on distribution P

Figure 1: The DAG m for our first example.

$0 \leq a,b,c,\ldots,g,h \leq 1$. Note that, due to Theorem 1, the equations in Table 1 fully determine P as a function of $a,b,c,\ldots,g,h$. Further, since (m,P) satisfies the Markov condition, each node is conditionally independent of the set of all its nondescendants given its parents. For example, we see that Z and U are conditionally independent given Y (written $Z \perp\!\!\!\perp U\,|\,Y$). Do these conditional independences entail any other conditional independences, that is, are there any other conditional independences which P must satisfy other than the one based on a node's parents? The answer is positive. For example, if (m,P) satisfies the Markov condition, then

$$P(w|u,y) = \sum_u P(w|z,u,y)P(z|u,y) = \sum_u P(w|z,y)P(z|y) = P(w|y)$$

and hence $W \perp\!\!\!\perp U\,|\,Y$. Explicitly, the notion of "entailed conditional independence" is given in the following definition:

Definition 2 *Let $m = (V,E)$ be a DAG where V is a set of random variables, and let $A,B,C \subset V$. We say that, based on Markov condition, m entails conditional independence $A \perp\!\!\!\perp B\,|\,C$ if $A \perp\!\!\!\perp B\,|\,C$ holds for every $P \in P_m$, where P_m is the set of all probability distributions P such that (m,P) satisfies the Markov condition.*

We say that there is a *direct dependency* between variables A and B in P if A and B are not conditionally independent given any subset of V. Based on the Markov condition, the absence of an edge between A and B implies that there is no direct dependency between A and B. However, the Markov condition is not sufficient to guarantee that the presence of an edge means direct dependency. In general, given a Bayesian network (m,P), we would want an edge in m to mean there is a direct dependency. In this case the DAG would become what it is naturally expected to be—a graphical representation of the structure of relationships between variables. The faithfulness condition as defined below indeed reflects this.

Definition 3 *We say that a Bayesian network (m,P) satisfies the faithfulness condition if, based on the Markov condition, m entails all and only the conditional independences in P. When (m,P)*

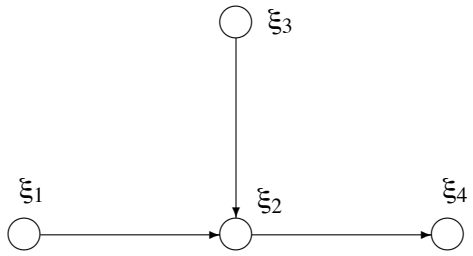


Figure 2: A second example of a DAG.

satisfies the faithfulness condition, we say that m and P are faithful to each other and we say m is a perfect map of P.

It is easy to see that the Bayesian network (m,P), where m is shown in Figure 1 and P satisfies the constraints in Table 1, does not satisfy the faithfulness condition. Indeed, Table 1 implies that $U \perp\!\!\!\perp Z$, but this independence is not entailed by m based on the Markov condition. As shown in Example 2.10 of Neapolitan (2004), the distribution P of this example has no perfect map. However, it is not hard to see that the DAG of Figure 1 is "optimal" in the sense that no DAG with fewer edges forms a Bayesian network with P.

In this paper we concentrate on Bayesian networks over a set of variables $X = \{\xi_1,\ldots,\xi_N\} \sim P$ where each variable takes values from the set $\{1,2\}$. Let m be a DAG with nodes $1,\ldots,N$. The probability distributions in P_m can be parameterized according to the conditional distributions of Theorem 1 as follows. For each node i, let $|\mathrm{Pa}(i)|$ be the number of parents of i and let $q_i(m) = 2^{|\mathrm{Pa}(i)|}$ be the number of possible states of the set of variables $\xi_{\mathrm{Pa}(i)}$. Consider a fixed list of the $q_i(m)$ possible states of $\xi_{\mathrm{Pa}(i)}$. For $j \in \{1,\ldots,q_i(m)\}$, we shall write "$\xi_{\mathrm{Pa}(i)} = j$" to mean that the parents of node i are in the states given by the j^{th} item in the list. For $k = 1,2$ and $j = 1,\ldots,q_i(m)$, we write $\theta_{ijk} = P(\xi_i = k \,|\, \xi_{\mathrm{Pa}(i)} = j)$. Observe that $\theta_{ij2} = 1 - \theta_{ij1}$. We shall write Θ_m to denote the vector of all θ_{ij1}'s for m:

$$\Theta_m \;=\; \big(\theta_{ij1} \,:\, i = 1,\ldots,N,\, j = 1,\ldots,q_i(m)\big) \;\in\; [0,1]^{k_m},$$

where $k_m = \sum_{i=1}^{N} q_i(m)$. Then each Θ_m in $[0,1]^{k_m}$ determines a probability measure $P = P_{\Theta_m}$ such that (m,P) is Bayesian network; and conversely, if (m,P) is a Bayesian network, then $P = P_{\Theta_m}$ for some $\Theta_m \in [0,1]^{k_m}$.

To illustrate this notation, consider the DAG m in Figure 2. Here, $\mathrm{Pa}(1) = \emptyset = \mathrm{Pa}(3)$, $\mathrm{Pa}(2) = \{1,3\}$, and $\mathrm{Pa}(4) = \{3\}$, and so $q_1(m) = 2^0 = q_3(m)$, $q_2(m) = 2^2$, and $q_4(m) = 2^1$. We could fix the list of possible states of $\xi_{\mathrm{Pa}(4)}$ to be "1,2", and the list for $\xi_{\mathrm{Pa}(2)}$ to be "(1,1), (1,2), (2,1), (2,2)" (with the understanding that the ordering is (ξ_1,ξ_3)). For the latter list, we have for example

$$\theta_{231} \;=\; P\big(\xi_2 = 1 \,|\, \xi_{\mathrm{Pa}(2)} = 3\big) \;=\; P\big(\xi_2 = 1 \,|\, (\xi_1,\xi_3) = (2,1)\big)\,.$$

Since $\mathrm{Pa}(3) = \emptyset$, $P(\xi_{\mathrm{Pa}(3)} = 1) = 1$, and θ_{311} is simply $P(\xi_3 = 1)$. We can write

$$\Theta_m \;=\; (\theta_{111},\theta_{211},\theta_{221},\theta_{231},\theta_{241},\theta_{311},\theta_{411},\theta_{421}),$$

and $k_m = 1+4+1+2 = 8$.

Let $D = D_n = \{X_1, ..., X_n\}$ be fully observed data of size n generated according to Pr, and let N_{ijk} be the number of cases in the database D such that node i takes value k while its parent set $\xi_{\mathrm{Pa}(i)}$ takes the values corresponding to j.

A *probabilistic model* $\mathcal{M}$ for a random vector $X = (\xi_1, \ldots, \xi_N)$ is a set of possible joint probability distributions of its components. If the probability distribution P is a member of a model $\mathcal{M}$, we say P is *included* in $\mathcal{M}$. Let m be a DAG (V,E). A *Bayesian network model* is a pair (m,F) where F is a set of possible parameter vectors Θ_m: each Θ_m in F determines conditional probability distributions for m, such that the joint probability distribution P_{Θ_m} of X (given by the product of these conditional distributions) satisfies the Markov condition with m. (E.g., for the DAG m of Figure 2, the most general choice of F is $[0,1]^8$, but F could also be a subset of $[0,1]^8$.) For simplicity, we shall usually omit F when referring to a Bayesian network model (m,F). In a given class of models, if $\mathcal{M}_2$ includes the probability distribution P, and if there exists no $\mathcal{M}_1$ (in the class) such that $\mathcal{M}_1$ includes P and $\mathcal{M}_1$ has smaller dimension than $\mathcal{M}_2$, then $\mathcal{M}_2$ is called a *parameter optimal map* of P. (E.g. the DAG of Figure 1 is a parameter optimal map of the distribution P of Table 1.) For the Bayesian network models we shall work with in this paper, the dimension of a model m is $k_m = \sum_{i=1}^{N} q_i(m)$. A detailed discussion of probabilistic model selection in the case of Bayesian networks could be found in Neapolitan (2004).

In order to proceed further we would also need a formal definition of consistency. In this definition we assume that the dimensions of the probabilistic models are well-defined. For a more detailed discussion of the definition of consistency see, for example, Neapolitan (2004), Grünwald (2007) and Lahiri (2001).

Definition 4 *Let D_n be a set of values (data) of a set of n mutually independent random vectors $X_1, \ldots, X_n$, each with probability distribution P. Furthermore, let* `score` *be a scoring criterion over some class of models for the random variables that constitute each vector. We say* `score` *is* ***consistent*** *for the class of models if the following two properties hold:*
1. If $\mathcal{M}_1$ includes P and $\mathcal{M}_2$ does not, then

$$\lim_{n\to\infty} Pr(\texttt{score}(D_n, \mathcal{M}_1) > \texttt{score}(D_n, \mathcal{M}_2)) = 1.$$

2. If $\mathcal{M}_1$ and $\mathcal{M}_2$ both include P and $\mathcal{M}_1$ has smaller dimension than $\mathcal{M}_2$, then

$$\lim_{n\to\infty} Pr(\texttt{score}(D_n, \mathcal{M}_1) > \texttt{score}(D_n, \mathcal{M}_2)) = 1.$$

Additionally, we say that the scoring criterion is ***strongly consistent*** *if, in both cases 1 and 2, it selects the appropriate model almost surely:*

$$Pr(\exists N : \forall n \geq N \;\; \texttt{score}(D_n, \mathcal{M}_1) > \texttt{score}(D_n, \mathcal{M}_2)) = 1.$$

As an example, let m_1 be the DAG of Figure 2, let m_2 be the DAG obtained from m_1 by adding an arc from node 3 to node 4, and let m_0 be the DAG obtained from m_1 by removing the arc from node 2 to node 4. For $i = 0,1,2$, let $\mathcal{M}_i$ be the probabilistic model consisting of all probability distributions with which m_i forms a Bayesian network. Let P be a probability distribution in $\mathcal{M}_1$ such the components of Θ_{m_1} are eight distinct numbers in $(0,1)$. Then $\mathcal{M}_0$ does not contain P (since ξ_4 is not independent of $\{\xi_1, \xi_2, \xi_3\}$), while $\mathcal{M}_1$ and $\mathcal{M}_2$ both contain P, and $\mathcal{M}_1$ has smaller

dimension that $\mathcal{M}_2$. If `score` is consistent, then in a situation with lots of data, `score` will be very likely to rank $\mathcal{M}_1$ over either $\mathcal{M}_0$ or $\mathcal{M}_2$. However, consider an infinite stream of data $X_1, X_2, \ldots$ sampled independently from P. Suppose that after each new observation, we ask `score` to choose among $\mathcal{M}_0$, $\mathcal{M}_1$ and $\mathcal{M}_2$. Consistency says that the expected proportion of `score`'s correct choices tends to 1 as n tends to infinity. But strong consistency says more: if `score` is strongly consistent, then with probability one it will make the correct choice for all but finitely many values of n.

3. Results

In this paper we consider the case of binomial Bayesian networks with independent $Beta(\alpha_{ij1}, \alpha_{ij2})$ priors for the parameters θ_{ij1} (note that $\theta_{ij2} = 1 - \theta_{ij1}$), where $\alpha_{ij1}, \alpha_{ij2} > 0$. We choose the beta family as it is the conjugate prior for the Binomial distribution. According to (1), the value of the Bayesian scoring criterion can be calculated as follows:

$$\begin{aligned} p(D_n|m) &= \int_{\Omega_m} p(D_n|m,\Theta_m)p(\Theta_m|m)d\Theta_m \\ &= \prod_{i=1}^{N}\prod_{j=1}^{q_i(m)} \int_0^1 \theta_{ij1}^{N_{ij1}+\alpha_{ij1}-1}(1-\theta_{ij1})^{N_{ij2}+\alpha_{ij2}-1}\frac{1}{Beta(\alpha_{ij1},\alpha_{ij2})}d\theta_{ij1} \\ &= \prod_{i=1}^{N}\prod_{j=1}^{q_i(m)} \frac{Beta(N_{ij1}+\alpha_{ij1}, N_{ij2}+\alpha_{ij2})}{Beta(\alpha_{ij1},\alpha_{ij2})} \\ &= \prod_{i=1}^{N}\prod_{j=1}^{q_i(m)} \frac{\Gamma(N_{ij1}+\alpha_{ij1})\Gamma(N_{ij2}+\alpha_{ij2})}{\Gamma(N_{ij1}+N_{ij2}+\alpha_{ij1}+\alpha_{ij2})}\cdot\frac{\Gamma(\alpha_{ij1}+\alpha_{ij2})}{\Gamma(\alpha_{ij1})\Gamma(\alpha_{ij2})}, \end{aligned} \quad (2)$$

which coincides with the well-known formula by Cooper and Herskovits (1992).

Throughout this paper we produce several asymptotic expansions "in probability" and "almost surely", always with respect to our probability measure Pr on Ω. We derive several properties of the marginal likelihood (2). We shall show that, for any model m,

$$\log p(D_n|m) = nC_m + \mathrm{O}(\sqrt{n\log\log n}) \qquad \text{a.s.}, \quad (3)$$

where C_m is a constant independent of n. We strengthen this result by showing how to obtain a positive constant σ_m such that

$$\limsup_{n\to\infty} \frac{\log p(D_n|m) - nC_m}{\sqrt{2n\mathrm{loglog}n}} = \sigma_m \quad \text{and} \quad \liminf_{n\to\infty} \frac{\log p(D_n|m) - nC_m}{\sqrt{2n\mathrm{loglog}n}} = -\sigma_m \quad \text{a.s.} \quad (4)$$

We note that "in probability" versions of the above statements also follow from our methods (as in the proofs of Corollaries 12 and 10):

$$\log p(D_n|m) \;=\; nC_m + O_p(\sqrt{n}), \quad (5)$$

$$\frac{\log p(D_n|m) - nC_m}{\sqrt{n}} \;\xrightarrow{\mathcal{D}}\; N(0,\sigma_m).$$

Additionally, we will be using the approximation of the Bayesian scoring criterion via maximum log-likelihood:

$$\log p(D_n|m) = \log\left(\prod_{i=1}^{N}\prod_{j=1}^{q_i(m)}\hat{\theta}_{ij1}^{N_{ij1}}(1-\hat{\theta}_{ij1})^{N_{ij2}}\right) - \frac{1}{2}k_m \log n + O_p(1), \tag{6}$$

where $\hat{\theta}_{ij1}$ is the MLE of θ_{ij1} and $k_m = \sum_{i=1}^{N} q_i(m)$ is the dimension of the model m. This is the efficient approximation of Bayesian score commonly known as BIC which was first derived in Schwarz (1978) for the case of linear exponential families. In Haughton (1988) his result was made more specific and extended to the case of curved exponential families—the type of model that includes Bayesian networks, as is shown in Geiger et al. (2001).

In this work, we attempt to get insight into the rate of convergence of the Bayes factor comparing two models m_1 and m_2. Our result strengthens the well known property of consistency of the Bayesian scoring criterion (e.g., see Chickering, 2002) and is expressed as the following theorem.

Theorem 5 *In the case of a binomial Bayesian network class, for the Bayesian scoring criterion based on independent beta priors, the following two properties hold:*

1. If m_2 includes P and m_1 does not, then there exists a positive constant $C(P,m_1,m_2)$ such that

$$\log\frac{\texttt{score}_B(D_n|m_2)}{\texttt{score}_B(D_n|m_1)} = C(P,m_1,m_2)n + O(\sqrt{n\log\log n}) \qquad a.s.$$

and

$$\log\frac{\texttt{score}_B(D_n|m_2)}{\texttt{score}_B(D_n|m_1)} = C(P,m_1,m_2)n + O_p(\sqrt{n}).$$

2. If m_1 and m_2 both include P and $\dim m_1 > \dim m_2$ where $\dim m_k = \sum_{i=1}^{N} q_i(m_k)$, $k = 1,2$, then

$$\log\frac{\texttt{score}_B(D_n|m_2)}{\texttt{score}_B(D_n|m_1)} = \frac{\dim m_1 - \dim m_2}{2}\log n + O(\log\log n) \qquad a.s.$$

and

$$\log\frac{\texttt{score}_B(D_n|m_2)}{\texttt{score}_B(D_n|m_1)} = \frac{\dim m_1 - \dim m_2}{2}\log n + O_p(1).$$

In particular, the Bayesian scoring criterion is strongly consistent.

It follows from the consistency property of the Bayesian scoring criterion that if P admits a faithful DAG representation, then the limit of the probability that a consistent scoring criterion chooses a model faithful to P, as the size of data approaches infinity, equals 1. Our result in Theorem 5 strengthens this claim as follows:

Corollary 6 *If (m_1,P) satisfies the faithfulness condition and (m_2,P) does not, then with probability 1, $\frac{\texttt{score}_B(D_n|m_1)}{\texttt{score}_B(D_n|m_2)}$ approaches infinity at exponential rate in n when m_2 does not include P, and approaches infinity at polynomial rate in n when m_2 includes P.*

The first result of Theorem 5 is optimal in the following sense:

Theorem 7 *If m_2 includes P and m_1 does not, then there exist $C_{m_1,m_2} > 0$ and $\sigma_{m_1,m_2} > 0$ such that:*

$$\limsup_{n\to\infty} \frac{\log \frac{\texttt{score}_B(D_n|m_2)}{\texttt{score}_B(D_n|m_1)} - nC_{m_1,m_2}}{\sigma_{m_1,m_2}\sqrt{2n\log\log n}} = 1 \qquad a.s.,$$

$$\liminf_{n\to\infty} \frac{\log \frac{\texttt{score}_B(D_n|m_2)}{\texttt{score}_B(D_n|m_1)} - nC_{m_1,m_2}}{\sigma_{m_1,m_2}\sqrt{2n\log\log n}} = -1 \qquad a.s.$$

and also

$$\frac{\log \frac{\texttt{score}_B(D_n|m_2)}{\texttt{score}_B(D_n|m_1)} - nC_{m_1,m_2}}{\sqrt{n}} \xrightarrow{\mathcal{D}} N(0,\sigma_{m_1,m_2}).$$

The constants C_{m_1,m_2} and σ_{m_1,m_2} from the above theorem could be defined as follows.

Definition 8 *Consider a single observation $X = (\xi_1,\ldots,\xi_N)$ from P. Define*

$$\phi_{ijk}(X) = \begin{cases} \log\theta_{ijk} & \text{if } \xi_i = k \text{ and } \xi_{Pa(i)} = j \\ 0 & \text{otherwise} \end{cases}$$

and define

$$\tau(X,m) = \sum_{i=1}^{N}\sum_{j=1}^{q_i(m)}\sum_{k=1}^{2}\phi_{ijk}(X).$$

Then define C_{m_1,m_2} and σ_{m_1,m_2} respectively to be the mean and the standard deviation of $\tau(X,m_1) - \tau(X,m_2)$. Also define

$$C_{i,m} = \prod_{j=1}^{q_i(m)}\left[\theta_{ij1}^{\theta_{ij1}}(1-\theta_{ij1})^{1-\theta_{ij1}}\right]^{P\left(\xi_{Pa(i)}=j\right)}.$$

Observe that we have

$$\tau(X,m) = \sum_{i=1}^{N}\log\theta_{i,\xi_{\mathrm{Pa}(i)},\xi_i}.$$

We shall show (see Lemma 13) that $\tau(X,m) = \log P(X)$, and (see proof of Lemma 9) that

$$C_{m_1,m_2} = \sum_{i=1}^{N}\log\frac{C_{i,m_2}}{C_{i,m_1}}. \tag{7}$$

Observe that for any i, the constant $C_{i,m}$ depends only on the conditional probabilities $P(\xi_i|\xi_{\mathrm{Pa}(i)})$ of the model m; therefore, if models m_1 and m_2 have the same set of parents of the i^{th} node, then $C_{i,m_1} = C_{i,m_2}$ and the i^{th} term in (7) is zero.

The quantities defined above will be extensively used throughout the Appendix.

4. Conclusion

In this paper we proved the strong consistency property of the Bayesian scoring criterion for the case of binomial Bayesian network models. We obtained asymptotic expansions for the logarithm of Bayesian score as well as the logarithm of the Bayes factor comparing two models. These results are important extensions of the consistency property of the Bayesian scoring criterion, providing insight into the rates at which the Bayes factor favors correct models. The asymptotic properties are found to be independent of the particular choice of beta parameter priors.

The methods we used are different from the mainstream. One typical way to investigate the properties of Bayesian score is to use BIC approximation and hence reduce the problem to investigation of the maximum log-likelihood term. In this paper we use expression (9) where the first term is the log-likelihood evaluated at the true parameter.

If we use the results of Theorem 5 in the approximation of Bayes scoring criterion by BIC (6), we can see that given two models m_1 and m_2, if both of them include the generating distribution P then their maximum log-likelihoods are within $O(\log\log n)$ of each other, and if one of the models does not include P then the maximum log-likelihoods differ by a leading order of $C(P,m_1,m_2)n$. These are the rates obtained by Qian and Field (2002, Theorems 2 and 3) for the case of model selection in logistic regression. This observation advocates for the existence of a unified approach for a very general class of models which can describe the rates at which Bayesian scoring criterion and its approximations favor correct model choices.

Acknowledgments

We would like to thank Hélène Massam, Yuehua Wu, Guoqi Qian, Chris Field, and the referees for their constructive and valuable comments and suggestions. A part of this work was done during the stay of the second author at the Institute of Mathematical Stochastics, University of Göttingen. He is thankful to M. Denker for his hospitality during this visit. This work was supported in part by a Discovery Grant from NSERC Canada (NM), Russian Foundation for Basic Research (DZ)(09-01-91331-NNIO-a, 09-01-00107-a), Russian Science Support Foundation (DZ) and Grant of the President of the Russian Federation (DZ)(NSh-638.2008.1).

Appendix A.

In this section we provide proofs for the basic facts in this paper and for Theorems 5 and 7. Note that part 1 of Theorem 5 follows directly from Theorem 7. In our derivations we first assume that the parameter prior of every node follows a flat $Beta(1,1)$ distribution. At the end, we shall show how the results can be extended to the case of general beta priors.

We will start from the expression for the marginal likelihood (2). Noticing that $Beta(x+1,y+1) = \left[(x+y+1)\binom{x+y}{x}\right]^{-1}$ we obtain the expression for the Bayesian scoring criterion via binomial coefficients:

$$p(D_n|m) = \left[\prod_{i=1}^{N}\prod_{j=1}^{q_i(m)}(N_{ij1}+N_{ij2}+1)\binom{N_{ij1}+N_{ij2}}{N_{ij1}}\right]^{-1}. \tag{8}$$

Let $P(k,n,\theta) = \binom{n}{k}\theta^k(1-\theta)^{n-k}$. Substituting $\binom{n}{k} = \frac{P(k,n,\theta)}{\theta^k(1-\theta)^{n-k}}$ into (8) with θ taken as θ_{ij1} we obtain an expression for $\log p(D_n|m)$, which will be the fundamental core of our proof:

$$\log p(D_n|m) = \sum_{i=1}^{N}\sum_{j=1}^{q_i(m)}\left[\log\left(\theta_{ij1}^{N_{ij1}}(1-\theta_{ij1})^{N_{ij2}}\right) - \log P(N_{ij1}, N_{ij1}+N_{ij2}, \theta_{ij1}) - \log(N_{ij1}+N_{ij2}+1)\right]. \quad (9)$$

The rest of the Appendix is organized as follows. In Lemma 9 we derive the law of iterated logarithm for the first term of (9) by using the function $\tau(X,m)$ introduced in Definition 8. Lemma 11 states asymptotic expansions of each of three terms in (9) hence providing us with an opportunity to get an expansion for $p(D_n|m)$. Asymptotic expressions (3), (4) and (5) are immediate consequences of this lemma. Lemma 13 establishes a fundamental result regarding the log-likelihood evaluated at the true parameter value. It is followed by the proofs of Theorems 5 and 7.

Lemma 9 *Recall the notation of Section 2 and Definition 8. For model m, let $T(m) = T_n(m) = \sum_{i=1}^{N}\sum_{j=1}^{q_i(m)}\log\left(\theta_{ij1}^{N_{ij1}}(1-\theta_{ij1})^{N_{ij2}}\right)$. Then the following laws of the iterated logarithm hold almost surely:*

$$\limsup_{n\to\infty}\frac{T(m) - n\sum_{i=1}^{N}\log C_{i,m}}{\sigma_m\sqrt{2n\log\log n}} = 1, \qquad \liminf_{n\to\infty}\frac{T(m) - n\sum_{i=1}^{N}\log C_{i,m}}{\sigma_m\sqrt{2n\log\log n}} = -1, \quad (10)$$

$$\limsup_{n\to\infty}\frac{[T(m_1) - T(m_2)] - nC_{m_1,m_2}}{\sigma_{m_1,m_2}\sqrt{2n\log\log n}} = 1, \qquad \liminf_{n\to\infty}\frac{[T(m_1) - T(m_2)] - nC_{m_1,m_2}}{\sigma_{m_1,m_2}\sqrt{2n\log\log n}} = -1. \quad (11)$$

Proof It is not difficult to see that $T(m) = \sum_{r=1}^{n}\tau(X_r,m)$ and

$$E(\tau(X,m)) = \sum_{i=1}^{N}\sum_{j=1}^{q_i(m)}\sum_{k=1}^{2} P\left(\xi_{\mathrm{Pa}(i)} = j\right)\theta_{ijk}\log\theta_{ijk} = \sum_{i=1}^{N}\log C_{i,m}.$$

By the law of the iterated logarithm applied to $T(m)$ we conclude that

$$\limsup_{n\to\infty}\frac{T(m) - n\sum_{i=1}^{N}\log C_{i,m}}{\sigma_m\sqrt{2n\log\log n}} = 1,$$

where σ_m is the standard deviation of $\tau(X,m)$. Further, applying the law of the iterated logarithm to $T(m_1) - T(m_2)$, we obtain the equalities (11) where σ_{m_1,m_2} is the standard deviation of $\tau(X,m_1) - \tau(X,m_2)$. ■

Corollary 10 *The following expressions hold:*

$$T(m_1) - T(m_2) = nC_{m_1,m_2} + O(\sqrt{n \log\log n}) \qquad a.s., \tag{12}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{q_i(m)} \log\left(\theta_{ij1}^{N_{ij1}} (1-\theta_{ij1})^{N_{ij2}}\right) = n \sum_{i=1}^{N} \log C_{i,m} + O_p(\sqrt{n}), \tag{13}$$

$$\frac{[T(m_1) - T(m_2)] - nC_{m_1,m_2}}{\sqrt{n}} \xrightarrow{\mathcal{D}} N(0, \sigma_{m_1,m_2}). \tag{14}$$

Proof Obviously, (12) is a direct consequence of (11). Applying the central limit theorem to $T(m)$ and $T(m_1) - T(m_2)$ in the proof of Lemma 9 instead of the law of the iterated logarithm we obtain (13) and (14). ■

Lemma 11 *The following asymptotic expansions hold:*

$$\sum_{i=1}^{N} \sum_{j=1}^{q_i(m)} \log\left(\theta_{ij1}^{N_{ij1}} (1-\theta_{ij1})^{N_{ij2}}\right) = n \sum_{i=1}^{N} \log C_{i,m} + O(\sqrt{n \log\log n}) \qquad a.s.,$$

$$N_{ij1} + N_{ij2} + 1 = n\left[P\left(\xi_{Pa(i)} = j\right) + o(1)\right] \qquad a.s., \tag{15}$$

$$\log P(N_{ij1}, N_{ij1} + N_{ij2}, \theta_{ij1}) = -\frac{1}{2} \log n + O(\log\log n) \qquad a.s., \tag{16}$$

$$\log P(N_{ij1}, N_{ij1} + N_{ij2}, \theta_{ij1}) = -\frac{1}{2} \log n + O_p(1). \tag{17}$$

Proof The first expression follows from (10). Further, note, that each of the variables N_{ijk} is a sum of i.i.d. Bernoulli variables. Based on the law of the iterated logarithm for the number of successes in n Bernoulli trials, as $n \to \infty$

$$N_{ijk} = n\theta_{ijk} P(\xi_{\mathrm{Pa}(i)} = j) + O(\sqrt{n \mathrm{loglog} n}) \qquad \text{a.s.}, \tag{18}$$

which immediately implies (15). Additionally, using the central limit theorem instead of the law of the iterated logarithm we obtain:

$$N_{ijk} = n\theta_{ijk} P(\xi_{\mathrm{Pa}(i)} = j) + O_p(\sqrt{n}). \tag{19}$$

Next, we will be using the following version of Local De Moivre-Laplace theorem (see for example p. 46 of Chow and Teicher 1978):

If $n \to \infty$ and $k = k_n \to \infty$ are such that $x_k n^{-\frac{1}{6}} \to 0$, where $x_k = \frac{k-np}{\sqrt{np(1-p)}}$, then

$$P(k,n,p) = \frac{1}{\sqrt{2\pi np(1-p)}} e^{-\frac{(k-np)^2}{2np(1-p)} + \mathrm{o}(1)}. \tag{20}$$

According to the law of the iterated logarithm, $x_k = \frac{k-np}{\sqrt{np(1-p)}} = \mathrm{O}(\sqrt{\log\log n})$ a.s., for the case where k is the number of successes in n i.i.d. Bernoulli trials of probability p. Notice that any such x_k satisfies the condition $x_k n^{-\frac{1}{6}} \to 0$. Therefore we can use (20) to approximate the binomial probability in (9), specifically:

$$\begin{aligned}&\log P(N_{ij1}, N_{ij1}+N_{ij2}, \theta_{ij1}) \\ &\quad = -\log\sqrt{2\pi(N_{ij1}+N_{ij2})\theta_{ij1}(1-\theta_{ij1})} - \frac{(N_{ij1}-(N_{ij1}+N_{ij2})\theta_{ij1})^2}{2(N_{ij1}+N_{ij2})\theta_{ij1}(1-\theta_{ij1})} + \mathrm{o}(1) \quad \text{a.s.} \quad (21)\end{aligned}$$

The first term in this expansion can be simplified based on (15):

$$-\log\sqrt{2\pi(N_{ij1}+N_{ij2})\theta_{ij1}(1-\theta_{ij1})} \;=\; -\frac{1}{2}\log n + O(1) \qquad \text{a.s.} \qquad (22)$$

Applying (18) to the second term we conclude that as $n \to \infty$:

$$-\frac{(N_{ij1}-(N_{ij1}+N_{ij2})\theta_{ij1})^2}{2(N_{ij1}+N_{ij2})\theta_{ij1}(1-\theta_{ij1})} \;=\; O(\log\log n) \qquad \text{a.s.} \qquad (23)$$

Now, (21) could be simplified further based on (22) and (23) to obtain (16). Finally, we can prove (17) analogously to (16) by using (19) instead of (18). Therefore the proof of the lemma is complete. ■

Now it is easy to derive the expansions announced in Sect. 3.

Corollary 12 *Properties (3), (4) and (5) of the marginal likelihood $P(D_n|m)$ hold.*

Proof Using (10), (15) and (16) in (9) and denoting $C_m \stackrel{def}{=} \sum_{i=1}^{N} \log C_{i,m}$ we get (4). Further, (3) is a direct consequence of (4). Finally, (5) can be proved by substituting (15), (13) and (16) into (9). ■

Lemma 13 *Suppose the probability distribution P is a member of the model m. Let $T(m) = \sum_{i=1}^{N}\sum_{j=1}^{q_i(m)} \log\left(\theta_{ij1}^{N_{ij1}}(1-\theta_{ij1})^{N_{ij2}}\right)$ for model m. Then $T(m) = \log P(D_n)$.*

Proof Since P is a member of the model m, we know that (m,P) satisfies the Markov condition. Therefore, by the factorization theorem (Theorem 1), we obtain

$$P(D_n) \;=\; \prod_{i=1}^{N}\prod_{j=1}^{q_i(m)} \theta_{ij1}^{N_{ij1}}(1-\theta_{ij1})^{N_{ij2}}$$

and the result follows. ■

Next, we will prove Theorems 5 and 7. Note that part 1 of Theorem 5 directly follows from Theorem 7.

Geiger et al. (2001) showed that each of the competing Bayesian network models m could be represented as a C^∞ connected manifold of dimension $\sum_{i=1}^N q_i(m)$ embedded in $\mathcal{R}^N$. In order to keep the notation simple we will be denoting this manifold as m. Every probability distribution for a finite sample space belongs to an exponential family. Therefore, there exists a set of ζ_i, $i = 1,2,\ldots,$ i.i.d. observations from an underlying distribution P_{θ_0} belonging to a full exponential family in standard form with densities $f(\zeta,\theta) = \exp(\zeta\theta - b(\theta))$ with respect to a finite measure on $\mathcal{R}^N$, with $\theta \in \Theta$ the natural parameter space, such that

$$\log\left(\prod_{i=1}^{N}\prod_{j=1}^{q_i(m)} \hat{\theta}_{ij1}^{N_{ij1}}(1-\hat{\theta}_{ij1})^{N_{ij2}}\right) = n \sup_{\phi\in m\cap\Theta}(Y_n\phi - b(\phi)), \tag{24}$$

where $Y_n = (1/n)\sum_{i=1}^n \zeta_i$.

Theorem 2.3 of Haughton (1988) provides an expansion of the logarithm of the Bayesian scoring criterion via maximum log-likelihood and, together with (24), guarantees (6). It follows from (3), (6) and (24) that

$$n \sup_{\phi\in m\cap\Theta}(Y_n\phi - b(\phi)) = C_m n + O_p(\sqrt{n\log\log n}). \tag{25}$$

Suppose m_2 includes P and m_1 does not. In this case, Haughton (1988, p.346) guarantees that as $n\to\infty$, we have

$$Pr\left(\sup_{\phi\in m_1\cap\Theta}(Y_n\phi - b(\phi)) + \varepsilon < \sup_{\phi\in m_2\cap\Theta}(Y_n\phi - b(\phi))\right) \to 1$$

for some $\varepsilon > 0$, and by (25) we obtain $C_{m_1} < C_{m_2}$. Hence, $\sum_{i=1}^N \log\frac{C_{i,m_2}}{C_{i,m_1}} > 0$. Now, the result of Theorem 7 can be obtained by using (15), (11) and (16) in (9), and by using (15), (14) and (16) in (9).

Now, suppose both m_1 and m_2 include the true distribution P and $k_{m_2} < k_{m_1}$. For part 2 of Theorem 5, direct application of Lemma 13, (16) and (15) provides the "almost surely" result, while Lemma 13, (17) and (15) prove the "in probability" result.

Finally, we shall show that the results of Theorems 5 and 7 hold for the case of general beta priors. It is not difficult to see that Stirling's approximation implies

$$\lim_{z\to\infty} z^{b-a}\frac{\Gamma(z+a)}{\Gamma(z+b)} = 1. \tag{26}$$

Denote as ψ_1 the flat $Beta(1,1)$ system of priors and denote as ψ_2 the system which, for each parameter θ_{ij1}, assumes the distribution $Beta(\alpha_{ij1},\alpha_{ij2})$, where $\alpha_{ij1},\alpha_{ij2} > 0$. It follows from (2) and (26) that:

$$\begin{aligned}
\frac{p(m|D,\psi_2)}{p(m|D,\psi_1)} &= \prod_{i=1}^{N}\prod_{j=1}^{q_i(m)} \frac{Beta(N_{ij1}+\alpha_{ij1}, N_{ij2}+\alpha_{ij2})}{Beta(N_{ij1}+1, N_{ij2}+1)} \cdot \frac{1}{Beta(\alpha_{ij1},\alpha_{ij2})} \\
&= \prod_{i=1}^{N}\prod_{j=1}^{q_i(m)} \frac{\Gamma(N_{ij1}+\alpha_{ij1})\Gamma(N_{ij2}+\alpha_{ij2})\Gamma(N_{ij1}+N_{ij2}+2)}{\Gamma(N_{ij1}+1)\Gamma(N_{ij2}+1)\Gamma(N_{ij1}+N_{ij2}+\alpha_{ij1}+\alpha_{ij2})} \cdot \frac{1}{Beta(\alpha_{ij1},\alpha_{ij2})} \\
&\sim \prod_{i=1}^{N}\prod_{j=1}^{q_i(m)} \frac{N_{ij1}^{\alpha_{ij1}-1} N_{ij2}^{\alpha_{ij2}-1}}{(N_{ij1}+N_{ij2})^{\alpha_{ij1}+\alpha_{ij2}-2}} \cdot \frac{1}{Beta(\alpha_{ij1},\alpha_{ij2})}.
\end{aligned}$$

Therefore, using (18) we can conclude that there exists a constant $c > 0$ such that:

$$\lim_{n\to\infty} \frac{p(m|D,\psi_2)}{p(m|D,\psi_1)} = c \quad \text{a.s.},$$

which implies that the results of Theorems 5 and 7 extend to the case of general beta parameter priors.

References

David M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

Yuan S. Chow and Henry Teicher. *Probability Theory: Independence, Interchangeability, Martingales*. Springer-Verlag, New York, 1978.

Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

Dan Geiger, David Heckerman, Henry King, and Christopher Meek. Stratified exponential families: Graphical models and model selection. *The Annals of Statistics*, 29(2):505–529, 2001.

Peter D. Grünwald. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

Dominique M. A. Haughton. On the choice of a model to fit data from an exponential family. *The Annals of Statistics*, 16(1):342–355, 1988.

Parhasarathi Lahiri. *Model selection*. Institute of Mathematical Statistics, Beachwood, Ohio, 2001.

Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2004.

Guoqi Qian and Chris Field. Law of iterated logarithm and consistent model selection criterion in logistic regression. *Statistics and Probability Letters*, 56(1):101, 2002.

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

Aad W. van der Vaart and Jon A. Wellner. *Weak Convergence and Empirical Processes*. Springer-Verlag, New York, 1996.

Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

Journal of Machine Learning Research 10 (2009) 1527-1570 Submitted 7/07; Revised 3/08; Published 7/09

Bayesian Network Structure Learning by Recursive Autonomy Identification

Raanan Yehezkel* RAANAN.YEHEZKEL@GMAIL.COM
Video Analytics Group
NICE Systems Ltd.
8 Hapnina, POB 690, Raanana, 43107, Israel

Boaz Lerner BOAZ@BGU.AC.IL
Department of Industrial Engineering and Management
Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel

Editor: Constantin Aliferis

Abstract

We propose the recursive autonomy identification (RAI) algorithm for constraint-based (CB) Bayesian network structure learning. The RAI algorithm learns the structure by sequential application of conditional independence (CI) tests, edge direction and structure decomposition into autonomous sub-structures. The sequence of operations is performed recursively for each autonomous sub-structure while simultaneously increasing the order of the CI test. While other CB algorithms d-separate structures and then direct the resulted undirected graph, the RAI algorithm combines the two processes from the outset and along the procedure. By this means and due to structure decomposition, learning a structure using RAI requires a smaller number of CI tests of high orders. This reduces the complexity and run-time of the algorithm and increases the accuracy by diminishing the curse-of-dimensionality. When the RAI algorithm learned structures from databases representing synthetic problems, known networks and natural problems, it demonstrated superiority with respect to computational complexity, run-time, structural correctness and classification accuracy over the PC, Three Phase Dependency Analysis, Optimal Reinsertion, greedy search, Greedy Equivalence Search, Sparse Candidate, and Max-Min Hill-Climbing algorithms.

Keywords: Bayesian networks, constraint-based structure learning

1. Introduction

A Bayesian network (BN) is a graphical model that efficiently encodes the joint probability distribution for a set of variables (Heckerman, 1995; Pearl, 1988). The BN consists of a structure and a set of parameters. The structure is a directed acyclic graph (DAG) that is composed of nodes representing domain variables and edges connecting these nodes. An edge manifests dependence between the nodes connected by the edge, while the absence of an edge demonstrates independence between the nodes. The parameters of a BN are conditional probabilities (densities) that quantify the graph edges. Once the BN structure has been learned, the parameters are usually estimated (in the case of discrete variables) using the relative frequencies of all combinations of variable states as exemplified in the data. Learning the structure from data by considering all possible structures ex-

*. This work was done while the author was at the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel.

haustively is not feasible in most domains, regardless of the size of the data (Chickering et al., 2004), since the number of possible structures grows exponentially with the number of nodes (Cooper and Herskovits, 1992). Hence, structure learning requires either sub-optimal heuristic search algorithms or algorithms that are optimal under certain assumptions.

One approach to structure learning—known as search-and-score (S&S) (Chickering, 2002; Cooper and Herskovits, 1992; Heckerman, 1995; Heckerman et al., 1995)—combines a strategy for searching through the space of possible structures with a scoring function measuring the fitness of each structure to the data. The structure achieving the highest score is then selected. Algorithms of this approach may also require node ordering, in which a parent node precedes a child node so as to narrow the search space (Cooper and Herskovits, 1992). In a second approach—known as constraint-based (CB) (Cheng et al., 1997; Pearl, 2000; Spirtes et al., 2000)—each structure edge is learned if meeting a constraint usually derived from comparing the value of a statistical or information-theory-based test of conditional independence (CI) to a threshold. Meeting such constraints enables the formation of an undirected graph, which is then further directed based on orientation rules (Pearl, 2000; Spirtes et al., 2000). That is, generally in the S&S approach we learn structures, whereas in the CB approach we learn edges composing a structure.

Search-and-score algorithms allow the incorporation of user knowledge through the use of prior probabilities over the structures and parameters (Heckerman et al., 1995). By considering several models altogether, the S&S approach may enhance inference and account better for model uncertainty (Heckerman et al., 1999). However, S&S algorithms are heuristic and usually have no proof of correctness (Cheng et al., 1997) (for a counter-example see Chickering, 2002, providing an S&S algorithm that identifies the optimal graph in the limit of a large sample and has a proof of correctness). As mentioned above, S&S algorithms may sometimes depend on node ordering (Cooper and Herskovits, 1992). Recently, it was shown that when applied to classification, a structure having a higher score does not necessarily provide a higher classification accuracy (Friedman et al., 1997; Grossman and Domingos, 2004; Kontkanen et al., 1999).

Algorithms of the CB approach are generally asymptotically correct (Cheng et al., 1997; Spirtes et al., 2000). They are relatively quick and have a well-defined stopping criterion (Dash and Druzdzel, 2003). However, they depend on the threshold selected for CI testing (Dash and Druzdzel, 1999) and may be unreliable in performing CI tests using large condition sets and a limited data size (Cooper and Herskovits, 1992; Heckerman et al., 1999; Spirtes et al., 2000). They can also be unstable in the sense that a CI test error may lead to a sequence of errors resulting in an erroneous graph (Dash and Druzdzel, 1999; Heckerman et al., 1999; Spirtes et al., 2000). Additional information on the above two approaches, their advantages and disadvantages, may be found in Cheng et al. (1997), Cooper and Herskovits (1992), Dash and Druzdzel (1999), Dash and Druzdzel (2003), Heckerman (1995), Heckerman et al. (1995), Heckerman et al. (1999), Pearl (2000) and Spirtes et al. (2000). We note that Cowell (2001) showed that for complete data, a given node ordering and using cross-entropy methods for checking CI and maximizing logarithmic scores to evaluate structures, the two approaches are equivalent. In addition, hybrid algorithms have been suggested in which a CB algorithm is employed to create an initial ordering (Singh and Valtorta, 1995), to obtain a starting graph (Spirtes and Meek, 1995; Tsamardinos et al., 2006a) or to narrow the search space (Dash and Druzdzel, 1999) for an S&S algorithm.

Most CB algorithms, such as Inductive Causation (IC) (Pearl, 2000), PC (Spirtes et al., 2000) and Three Phase Dependency Analysis (TPDA) (Cheng et al., 1997), construct a DAG in two consecutive stages. The first stage is learning associations between variables for constructing an undi-

rected structure. This requires a number of CI tests growing exponentially with the number of nodes. This complexity is reduced in the PC algorithm to polynomial complexity by fixing the maximal number of parents a node can have and in the TPDA algorithm by measuring the strengths of the independences computed while CI testing along with making a strong assumption about the underlying graph (Cheng et al., 1997). The TPDA algorithm does not take direct steps to restrict the size of the condition set employed in CI testing in order to mitigate the curse-of-dimensionality.

In the second stage, most CB algorithms direct edges by employing orientation rules in two consecutive steps: finding and directing V-structures and directing additional edges inductively (Pearl, 2000). Edge direction (orientation) is unstable. This means that small errors in the input to the stage (i.e., CI testing) yield large errors in the output (Spirtes et al., 2000). Errors in CI testing are usually the result of large condition sets. These sets, selected based on previous CI test results, are more likely to be incorrect due to their size, and they also lead, for a small sample size, to poorer estimation of dependences due to the curse-of-dimensionality. Thus, we usually start learning using CI tests of low order (i.e., using small condition sets), which are the most reliable tests (Spirtes et al., 2000). We further note that the division of learning in CB algorithms into two consecutive stages is mainly for simplicity, since no directionality constraints have to be propagated during the first stage. However, errors in CI testing is a main reason for the instability of CB algorithms, which we set out to tackle in this research.

We propose the recursive autonomy identification (RAI) algorithm, which is a CB model that learns the structure of a BN by sequential application of CI tests, edge direction and structure decomposition into autonomous sub-structures that comply with the Markov property (i.e., the sub-structure includes all its nodes' parents). This sequence of operations is performed recursively for each autonomous sub-structure. In each recursive call of the algorithm, the order of the CI test is increased similarly to the PC algorithm (Spirtes et al., 2000). By performing CI tests of low order (i.e., tests employing small conditions sets) before those of high order, the RAI algorithm performs more reliable tests first, and thereby obviates the need to perform less reliable tests later. By directing edges while testing conditional independence, the RAI algorithm can consider parent-child relations so as to rule out nodes from condition sets and thereby to avoid unnecessary CI tests and to perform tests using smaller condition sets. CI tests using small condition sets are faster to implement and more accurate than those using large sets. By decomposing the graph into autonomous sub-structures, further elimination of both the number of CI tests and size of condition sets is obtained. Graph decomposition also aids in subsequent iterations to direct additional edges. By recursively repeating both mechanisms for autonomies decomposed from the graph, further reduction of computational complexity, database queries and structural errors in subsequent iterations is achieved. Overall, the RAI algorithm learns faster a more precise structure.

Tested using synthetic databases, nineteen known networks, and nineteen UCI databases, RAI showed in this study superiority with respect to structural correctness, complexity, run-time and classification accuracy over PC, Three Phase Dependency Analysis, Optimal Reinsertion, a greedy hill-climbing search algorithm with a Tabu list, Greedy Equivalence Search, Sparse Candidate, naive Bayesian, and Max-Min Hill-Climbing algorithms.

After providing some preliminaries and definitions in Section 2, we introduce the RAI algorithm and prove its correctness in Section 3. Section 4 presents experimental evaluation of the RAI algorithm with respect to structural correctness, complexity, run-time and classification accuracy in comparison to CB, S&S and hybrid structure learning algorithms. Section 5 concludes the paper with a discussion.

2. Preliminaries

A BN $B(\mathcal{G},\Theta)$ is a model for representing the joint probability distribution for a set of variables $\boldsymbol{X} = \{X_1 \ldots X_n\}$. The structure $\mathcal{G}(\boldsymbol{V},\boldsymbol{E})$ is a DAG composed of $\boldsymbol{V}$, a set of nodes representing the domain variables $\boldsymbol{X}$, and $\boldsymbol{E}$, a set of directed edges connecting the nodes. A directed edge $X_i \rightarrow X_j$ connects a child node X_j to its parent node X_i. We denote $\boldsymbol{Pa}(X,\mathcal{G})$ as the set of parents of node X in a graph $\mathcal{G}$. The set of parameters Θ holds local conditional probabilities over $\boldsymbol{X}$, $P(X_i|\boldsymbol{Pa}(X_i,\mathcal{G}))\forall i$ that quantify the graph edges. The joint probability distribution for $\boldsymbol{X}$ represented by a BN that is assumed to encode this distribution[1] is (Cooper and Herskovits, 1992; Heckerman, 1995; Pearl, 1988)

$$P(X_1 \ldots X_n) = \prod_{i=1}^{n} P(X_i|\boldsymbol{Pa}(X_i,\mathcal{G})). \tag{1}$$

Though there is no theoretical restriction on the functional form of the conditional probability distributions in Equation 1, we restrict ourselves in this study to discrete variables. This implies joint distributions which are unrestricted discrete distributions and conditional probability distributions which are independent multinomials for each variable and each parent configuration (Chickering, 2002).

We also make use of the term partially directed graph, that is, a graph that may have both directed and undirected edges and has at most one edge between any pair of nodes (Meek, 1995). We use this term while learning a graph starting from a complete undirected graph and removing and directing edges until uncovering a graph representing a family of Markov equivalent structures (pattern) of the true underlying BN[2] (Pearl, 2000; Spirtes et al., 2000). $\boldsymbol{Pa}_p(X,\mathcal{G})$, $\boldsymbol{Adj}(X,\mathcal{G})$ and $\boldsymbol{Ch}(X,\mathcal{G})$ are, respectively, the sets of potential parents, adjacent nodes[3] and children of node X in a partially directed graph $\mathcal{G}$, $\boldsymbol{Pa}_p(X,\mathcal{G}) = \boldsymbol{Adj}(X,\mathcal{G}) \backslash \boldsymbol{Ch}(X,\mathcal{G})$.

We indicate that X and Y are independent conditioned on a set of nodes $\boldsymbol{S}$ (i.e., the condition set) using $X \perp\!\!\!\perp Y \,|\, \boldsymbol{S}$, and make use of the notion of d-separation (Pearl, 1988). Thereafter, we define d-separation resolution with the aim to evaluate d-separation for different sizes of condition sets, d-separation resolution of a graph, an exogenous cause to a graph and an autonomous substructure. We concentrate in this section only on terms and definitions that are directly relevant to the RAI concept and algorithm, where other more general terms and definitions relevant to BNs can be found in Heckerman (1995), Pearl (1988), Pearl (2000), and Spirtes et al. (2000).

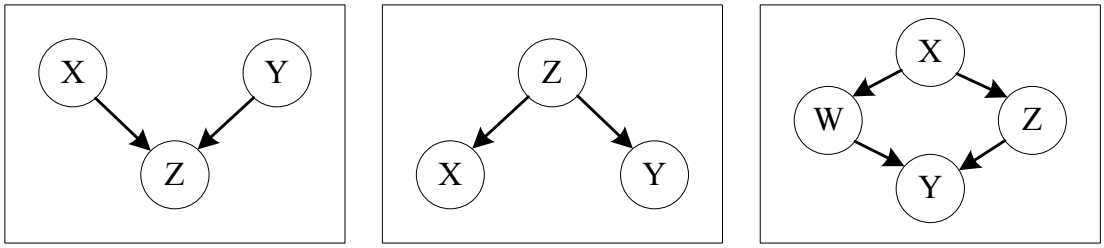
Definition 1 – *d-separation resolution*: The resolution of a d-separation relation between a pair of non-adjacent nodes in a graph is the size of the smallest condition set that d-separates the two nodes.

Examples of d-separation resolutions of 0, 1 and 2 between nodes X and Y are given in Figure 1.

Definition 2 – *d-separation resolution of a graph*: The d-separation resolution of a graph is the highest d-separation resolution in the graph.

The d-separation relations encoded by the example graph in Figure 2a and relevant to the determination of the d-separation resolution of this graph are: 1) $X_1 \perp\!\!\!\perp X_2 \,|\, \emptyset$; 2) $X_1 \perp\!\!\!\perp X_4 \,|\, \{X_3\}$; 3) $X_1 \perp\!\!\!\perp X_5 \,|\, \{X_3\}$; 4) $X_1 \perp\!\!\!\perp X_6 \,|\, \{X_3\}$; 5) $X_2 \perp\!\!\!\perp X_4 \,|\, \{X_3\}$; 6) $X_2 \perp\!\!\!\perp X_5 \,|\, \{X_3\}$; 7) $X_2 \perp\!\!\!\perp X_6 \,|\, \{X_3\}$; 8) $X_3 \perp\!\!\!\perp X_6 \,|\, \{X_4, X_5\}$ and 9) $X_4 \perp\!\!\!\perp X_5 \,|\, \{X_3\}$. Due to relation 8, exemplifying d-separation resolution

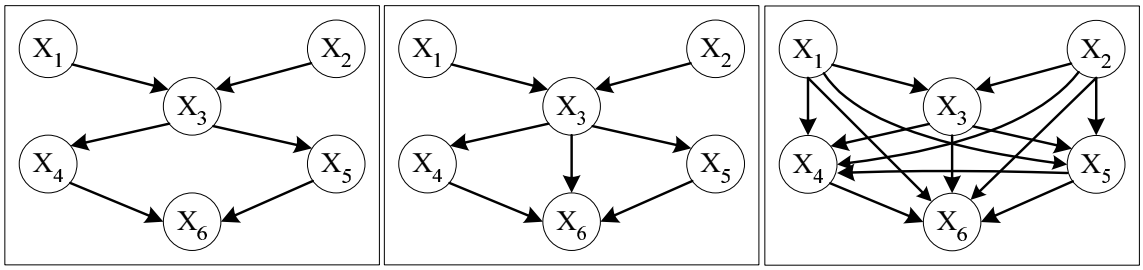
1. Throughout the paper, we assume faithfulness of the probability distribution to a DAG (Spirtes et al., 2000).
2. Two BNs are Markov equivalent if and only if they have the same sets of adjacencies and V-structures (Verma and Pearl, 1990).
3. Two nodes in a graph that are connected by an edge are adjacent.



(a) (b) (c)

Figure 1: Examples of d-separation resolutions of (a) 0, (b) 1 and (c) 2 between nodes X and Y.

of 2, the d-separation resolution of the graph is 2. Eliminating relation 8 by adding the edge $X_3 \rightarrow X_6$, we form a graph having a d-separation resolution of 1 (Figure 2b). By further adding edges to the graph, eliminating relations of resolution 1, we form a graph having a d-separation resolution of 0 (Figure 2c) that encodes only relation 1.



(a) (b) (c)

Figure 2: Examples of graph d-separation resolutions of (a) 2, (b) 1 and (c) 0.

Definition 3 – *exogenous cause*: A node Y in $\mathcal{G}(\boldsymbol{V},\boldsymbol{E})$ is an exogenous cause to $\mathcal{G}'(\boldsymbol{V}',\boldsymbol{E}')$, where $\boldsymbol{V}' \subset \boldsymbol{V}$ and $\boldsymbol{E}' \subset \boldsymbol{E}$, if $Y \notin \boldsymbol{V}'$ and $\forall X \in \boldsymbol{V}', Y \in \boldsymbol{Pa}(X,\mathcal{G})$ or $Y \notin \boldsymbol{Adj}(X,\mathcal{G})$ (Pearl, 2000).

Definition 4 – *autonomous sub-structure*: In a DAG $\mathcal{G}(\boldsymbol{V},\boldsymbol{E})$, a sub-structure $\mathcal{G}^{\mathrm{A}}(\boldsymbol{V}^{\mathrm{A}},\boldsymbol{E}^{\mathrm{A}})$ such that $\boldsymbol{V}^{\mathrm{A}} \subset \boldsymbol{V}$ and $\boldsymbol{E}^{\mathrm{A}} \subset \boldsymbol{E}$ is said to be autonomous in $\mathcal{G}$ given a set $\boldsymbol{V}_{ex} \subset \boldsymbol{V}$ of exogenous causes to $\mathcal{G}^{\mathrm{A}}$ if $\forall X \in \boldsymbol{V}^{\mathrm{A}}, \boldsymbol{Pa}(X,\mathcal{G}) \subset \{\boldsymbol{V}^{\mathrm{A}} \cup \boldsymbol{V}_{ex}\}$. If $\boldsymbol{V}_{ex}$ is empty, we say the sub-structure is (completely) autonomous[4].

We define sub-structure autonomy in the sense that the sub-structure holds the Markov property for its nodes. Given a structure $\mathcal{G}$, any two non-adjacent nodes in an autonomous sub-structure $\mathcal{G}^{\mathrm{A}}$ in $\mathcal{G}$ are d-separated given nodes either included in the sub-structure $\mathcal{G}^{\mathrm{A}}$ or exogenous causes to $\mathcal{G}^{\mathrm{A}}$. Figure 3 depicts a structure $\mathcal{G}$ containing a sub-structure $\mathcal{G}^{\mathrm{A}}$. Since nodes X_1 and X_2 are exogenous causes to $\mathcal{G}^{\mathrm{A}}$ (i.e., they are either parents of nodes in $\mathcal{G}^{\mathrm{A}}$ or not adjacent to them; see Definition 3), $\mathcal{G}^{\mathrm{A}}$ is said to be autonomous in $\mathcal{G}$ given nodes X_1 and X_2.

Proposition 1: If $\mathcal{G}^{\mathrm{A}}(\boldsymbol{V}^{\mathrm{A}},\boldsymbol{E}^{\mathrm{A}})$ is an autonomous sub-structure in a DAG $\mathcal{G}(\boldsymbol{V},\boldsymbol{E})$ given a set $\boldsymbol{V}_{ex} \subset \boldsymbol{V}$ of exogenous causes to $\mathcal{G}^{\mathrm{A}}$ and $X \perp\!\!\!\perp Y \,|\, \boldsymbol{S}$, where $X, Y \in \boldsymbol{V}^{\mathrm{A}}$, $\boldsymbol{S} \subset \boldsymbol{V}$, then $\exists \boldsymbol{S}'$ such that $\boldsymbol{S}' \subset \{\boldsymbol{V}^{\mathrm{A}} \cup \boldsymbol{V}_{ex}\}$ and $X \perp\!\!\!\perp Y \,|\, \boldsymbol{S}'$.

4. If $\mathcal{G}$ is a partially directed graph, then $\boldsymbol{Pa}_p(X,\mathcal{G})$ replaces $\boldsymbol{Pa}(X,\mathcal{G})$.

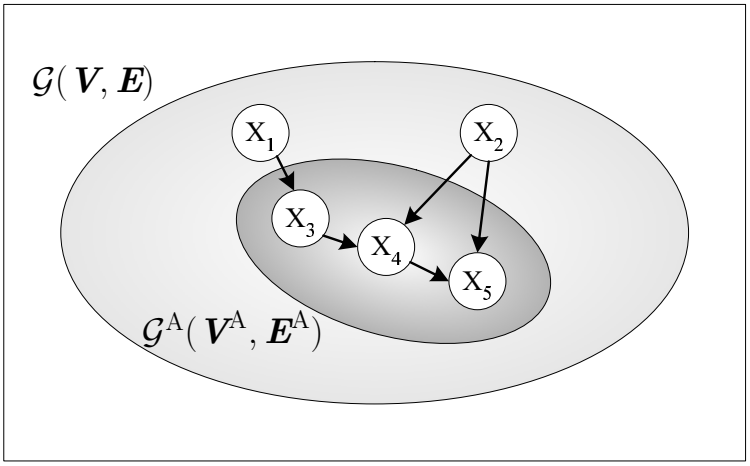


Figure 3: An example of an autonomous sub-structure.

Proof: The proof is based on Lemma 1.
Lemma 1: If in a DAG, X and Y are non-adjacent and X is not a descendant of Y,[5] then X and Y are d-separated given $\boldsymbol{Pa}(Y)$ (Pearl, 1988; Spirtes et al., 2000).

If in a DAG $\mathcal{G}(\boldsymbol{V},\boldsymbol{E})$, $X \perp\!\!\!\perp Y \,|\, \boldsymbol{S}$ for some set $\boldsymbol{S}$, where X and Y are non-adjacent, and if X is not a descendant of Y, then, according to Lemma 1, X and Y are d-separated given $\boldsymbol{Pa}(Y)$. Since X and Y are contained in the sub-structure $\mathcal{G}^{A}(\boldsymbol{V}^{A},\boldsymbol{E}^{A})$, which is autonomous given the set of nodes $\boldsymbol{V}_{ex}$, then, following the definition of an autonomous sub-structure, all parents of the nodes in $\boldsymbol{V}^{A}$—and specifically $\boldsymbol{Pa}(Y)$—are members in set $\{\boldsymbol{V}^{A} \cup \boldsymbol{V}_{ex}\}$. Then, $\exists \boldsymbol{S}'$ such that $\boldsymbol{S}' \subset \{\boldsymbol{V}^{A} \cup \boldsymbol{V}_{ex}\}$ and $X \perp\!\!\!\perp Y \,|\, \boldsymbol{S}'$, which proves Proposition 1. ■

3. Recursive Autonomy Identification

Starting from a complete undirected graph and proceeding from low to high graph d-separation resolution, the RAI algorithm uncovers the correct pattern[6] of a structure by performing the following sequence of operations: (1) test of CI between nodes, followed by the removal of edges related to independences, (2) edge direction according to orientation rules, and (3) graph decomposition into autonomous sub-structures. For each autonomous sub-structure, the RAI algorithm is applied recursively, while increasing the order of CI testing.

CI testing of order n between nodes X and Y is performed by thresholding the value of a criterion that measures the dependence between the nodes conditioned on a set of n nodes (i.e., the condition set) from the parents of X or Y. The set is determined by the Markov property (Pearl, 2000), for example, if X is directed into Y, then only Y's parents are included in the set. Commonly, this criterion is the χ^2 goodness of fit test (Spirtes et al., 2000) or conditional mutual information (CMI) (Cheng et al., 1997).

5. If X is a descendant of Y, we change the roles of X and Y and replace $\boldsymbol{Pa}(Y)$ with $\boldsymbol{Pa}(X)$.
6. In the absence of a topological node ordering, uncovering the correct pattern is the ultimate goal of BN structure learning algorithms, since a pattern represents the same set of probabilities as that of the true structure (Spirtes et al., 2000).

Directing edges is conducted according to orientation rules (Pearl, 2000; Spirtes et al., 2000). Given an undirected graph and a set of independences, both being the result of CI testing, the following two steps are performed consecutively. First, intransitive triplets of nodes (V-structures) are identified, and the corresponding edges are directed. An intransitive triplet $X \rightarrow Z \leftarrow Y$ is defined if 1) X and Y are non-adjacent neighbors of Z, and 2) Z is not in the condition set that separated X and Y. In the second step, also known as the inductive stage, edges are continually directed until no more edges can be directed, while assuring that no new V-structures and no directed cycles are created.

Decomposition into separated, smaller, autonomous sub-structures reveals the structure hierarchy. Decomposition also decreases the number and length of paths between nodes that are CI-tested, thereby diminishing, respectively, the number of CI tests and the sizes of condition sets used in these tests. Both reduce computational complexity. Moreover, due to decomposition, additional edges can be directed, which reduces the complexity of CI testing of the subsequent iterations. Following decomposition, the RAI algorithm identifies ancestor and descendant sub-structures; the former are autonomous, and the latter are autonomous given nodes of the former.

3.1 The RAI Algorithm

Similarly to other algorithms of structure learning (Cheng et al., 1997; Cooper and Herskovits, 1992; Heckerman, 1995), the RAI algorithm[7] assumes that all the independences entailed from the given data can be encoded by a DAG. Similarly to other CB algorithms of structure learning (Cheng et al., 1997; Spirtes et al., 2000), the RAI algorithm assumes that the data sample size is large enough for reliable CI tests.

An iteration of the RAI algorithm starts with knowledge produced in the previous iteration and the current d-separation resolution, n. Previous knowledge includes $\mathcal{G}_{\text{start}}$, a structure having a d-separation resolution of $n-1$, and $\mathcal{G}_{\text{ex}}$, a set of structures each having possible exogenous causes to $\mathcal{G}_{\text{start}}$. Another input is the graph $\mathcal{G}_{\text{all}}$, which contains $\mathcal{G}_{\text{start}}$, $\mathcal{G}_{\text{ex}}$ and edges connecting them. Note that $\mathcal{G}_{\text{all}}$ may also contain other nodes and edges, which may not be required for the learning task (e.g., edges directed from nodes in $\mathcal{G}_{\text{start}}$ into nodes that are not in $\mathcal{G}_{\text{start}}$ or $\mathcal{G}_{\text{ex}}$), and these will be ignored by the RAI. In the first iteration, $n=0$, $\mathcal{G}_{\text{ex}}=\emptyset$, $\mathcal{G}_{\text{start}}(\boldsymbol{V},\boldsymbol{E})$ is the complete undirected graph and the d-separation resolution is not defined, since there are no pairs of d-separated nodes. Since $\mathcal{G}_{\text{ex}}$ is empty, $\mathcal{G}_{\text{all}}=\mathcal{G}_{\text{start}}$.

Given a structure $\mathcal{G}_{\text{start}}$ having d-separation resolution $n-1$, the RAI algorithm seeks independences between adjacent nodes conditioned on sets of size n and removes the edges corresponding to these independences. The resulting structure has a d-separation resolution of n. After applying orientation rules so as to direct the remaining edges, a partial topological order is obtained in which parent nodes precede their descendants. Childless nodes have the lowest topological order. This order is partial, since not all the edges can be directed; thus, edges that cannot be directed connect nodes of equal topological order. Using this partial topological ordering, the algorithm decomposes the structure into ancestor and descendent autonomous sub-structures so as to reduce the complexity of the successive stages.

First, descendant sub-structures are established containing the lowest topological order nodes. A descendant sub-structure may be composed of a single childless node or several adjacent childless

7. The RAI algorithm and a preliminary experimental evaluation of the algorithm were introduced in Yehezkel and Lerner (2005).

nodes. We will further refer to a single descendent sub-structure, although such a sub-structure may consist of several non-connected sub-structures. Second, all edges pointing towards nodes of the descendant sub-structure are temporarily removed (together with the descendant sub-structure itself), and the remaining clusters of connected nodes are identified as ancestor sub-structures. The descendent sub-structure is autonomous, given nodes of higher topological order composing the ancestor sub-structures. To consider smaller numbers of parents (and thereby smaller condition set sizes) when CI testing nodes of the descendant sub-structure, the algorithm first learns ancestor sub-structures, then the connections between ancestor and descendant sub-structures, and finally the descendant sub-structure itself. Each ancestor or descendent sub-structure is further learned by recursive calls to the algorithm. Figures 4, 5 and 6 show, respectively, the RAI algorithm, a manifesting example and the algorithm execution order for this example.

The RAI algorithm is composed of four stages (denoted in Figure 4 as Stages A, B, C and D) and an exit condition checked before the execution of any of the stages. The purpose of the exit condition is to assure that a CI test of a required order can indeed be performed, that is, the number of potential parents required to perform the test is adequate. The purpose of Stage A1 is to thin the link between $\mathcal{G}_{\text{ex}}$ and $\mathcal{G}_{\text{start}}$, the latter having d-separation resolution of $n-1$. This is achieved by removing edges corresponding to independences between nodes in $\mathcal{G}_{\text{ex}}$ and nodes in $\mathcal{G}_{\text{start}}$ conditioned on sets of size n of nodes that are either exogenous to, or within, $\mathcal{G}_{\text{start}}$. Similarly, in Stage B1, the algorithm tests for CI of order n between nodes in $\mathcal{G}_{\text{start}}$ given sets of size n of nodes that are either exogenous to, or within, $\mathcal{G}_{\text{start}}$, and removes edges corresponding to independences. The edges removed in Stages A1 and B1 could not have been removed in previous applications of these stages using condition sets of lower orders. When testing independence between X and Y, conditioned on the potential parents of node X, those nodes in the condition set that are exogenous to $\mathcal{G}_{\text{start}}$ are X's parents whereas those nodes that are in $\mathcal{G}_{\text{start}}$ are either its parents or adjacents.

In Stages A2 and B2, the algorithm directs every edge from the remaining edges that can be directed. In Stage B3, the algorithm groups in a descendant sub-structure all the nodes having the lowest topological order in the derived partially directed structure, and following the temporary removal of these nodes, it defines in Stage B4 separate ancestor sub-structures. Due to the topological order, every edge from a node X in an ancestor sub-structure to a node Z in the descendant sub-structure is directed as $X \rightarrow Z$. In addition, there is no edge connecting one ancestor sub-structure to another ancestor sub-structure.

Thus, every ancestor sub-structure contains all the potential parents of its nodes, that is, it is autonomous (or if some potential parents are exogenous, then the sub-structure is autonomous given the set of exogenous nodes). The descendant sub-structure is, by definition, autonomous given nodes of ancestor sub-structures. Proposition 1 showed that we can identify all the conditional independences between nodes of an autonomous sub-structure. Hence, every ancestor and descendant sub-structure can be processed independently in Stages C and D, respectively, so as to identify conditional independences of increasing orders in each recursive call of the algorithm. Stage C is a recursive call for the RAI algorithm for learning each ancestor sub-structure with order $n+1$. Similarly, Stage D is a recursive call for the RAI algorithm for learning the descendant sub-structure with order $n+1$, while assuming that the ancestor sub-structures have been fully learned (having d-separation resolution of $n+1$).

Figure 5 and Figure 6, respectively, show diagrammatically the stages in learning an example graph and the execution order of the algorithm for this example. Figure 5a shows the true structure that we wish to uncover. Initially, $\mathcal{G}_{\text{start}}$ is the complete undirected graph (Figure 5b), $n=0$, $\mathcal{G}_{\text{ex}}$ is

Main function: $\mathcal{G}_{\text{out}} = \text{RAI}[n, \mathcal{G}_{\text{start}}(\boldsymbol{V}_{\text{start}}, \boldsymbol{E}_{\text{start}}), \mathcal{G}_{\text{ex}}(\boldsymbol{V}_{ex}, \boldsymbol{E}_{ex}), \mathcal{G}_{\text{all}}]$

Exit condition

If all nodes in $\mathcal{G}_{\text{start}}$ have fewer than $n+1$ potential parents, set $\mathcal{G}_{\text{out}} = \mathcal{G}_{\text{all}}$ and exit.

A. **Thinning the link between $\mathcal{G}_{\text{ex}}$ and $\mathcal{G}_{\text{start}}$ and directing $\mathcal{G}_{\text{start}}$**

1. For every node Y in $\mathcal{G}_{\text{start}}$ and its parent X in $\mathcal{G}_{\text{ex}}$, if $\exists \boldsymbol{S} \subset \{\boldsymbol{Pa}_p(Y, \mathcal{G}_{\text{start}}) \cup \boldsymbol{Pa}(Y, \mathcal{G}_{\text{ex}}) \backslash X\}$ and $|\boldsymbol{S}| = n$ such that $X \perp\!\!\!\perp Y \,|\, \boldsymbol{S}$, then remove the edge between X and Y from $\mathcal{G}_{\text{all}}$.
2. Direct the edges in $\mathcal{G}_{\text{start}}$ using orientation rules.

B. **Thinning, directing and decomposing $\mathcal{G}_{\text{start}}$**

1. For every node Y and its potential parent X both in $\mathcal{G}_{\text{start}}$, if $\exists \boldsymbol{S} \subset \{\boldsymbol{Pa}(Y, \mathcal{G}_{\text{ex}}) \cup \boldsymbol{Pa}_p(Y, \mathcal{G}_{\text{start}}) \backslash X\}$ and $|\boldsymbol{S}| = n$ such that $X \perp\!\!\!\perp Y \,|\, \boldsymbol{S}$, then remove the edge between X and Y from $\mathcal{G}_{\text{all}}$ and $\mathcal{G}_{\text{start}}$.
2. Direct the edges in $\mathcal{G}_{\text{start}}$ using orientation rules.
3. Group the nodes having the lowest topological order into a descendant sub-structure $\mathcal{G}_{\text{D}}$.
4. Remove $\mathcal{G}_{\text{D}}$ from $\mathcal{G}_{\text{start}}$ temporarily and define the resulting unconnected structures as ancestor sub-structures $\mathcal{G}_{\text{A}_1}, \ldots, \mathcal{G}_{\text{A}_k}$.

C. **Ancestor sub-structure decomposition**

For $i = 1$ to k, call $\text{RAI}[n+1, \mathcal{G}_{\text{A}_i}, \mathcal{G}_{\text{ex}}, \mathcal{G}_{\text{all}}]$.

D. **Descendant sub-structure decomposition**

1. Define $\mathcal{G}_{\text{ex}_{\text{D}}} = \{\mathcal{G}_{\text{A}_1}, \ldots, \mathcal{G}_{\text{A}_k}, \mathcal{G}_{\text{ex}}\}$ as the exogenous set to $\mathcal{G}_{\text{D}}$.
2. Call $\text{RAI}[n+1, \mathcal{G}_{\text{D}}, \mathcal{G}_{\text{ex}_{\text{D}}}, \mathcal{G}_{\text{all}}]$.
3. Set $\mathcal{G}_{\text{out}} = \mathcal{G}_{\text{all}}$ and exit.

Figure 4: The RAI algorithm.

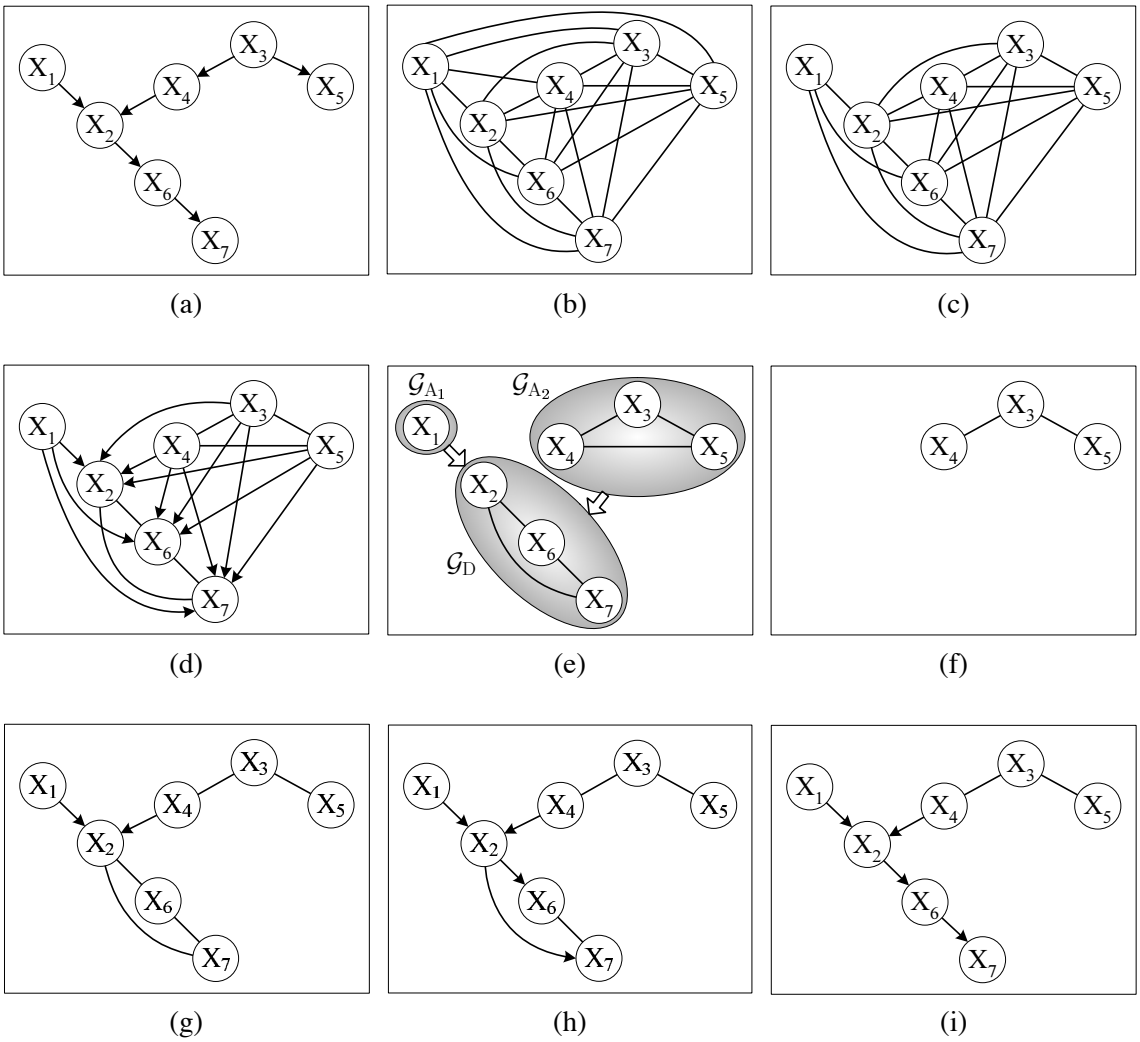


Figure 5: Learning an example structure. a) The true structure to learn, b) initial (complete) structure and structures learned by the RAI algorithm in Stages (see Figure 4) c) B1, d) B2, e) B3 and B4, f) C, g) D and A1, h) D and A2 and i) D, B1 and B2 (i.e., the resulting structure).

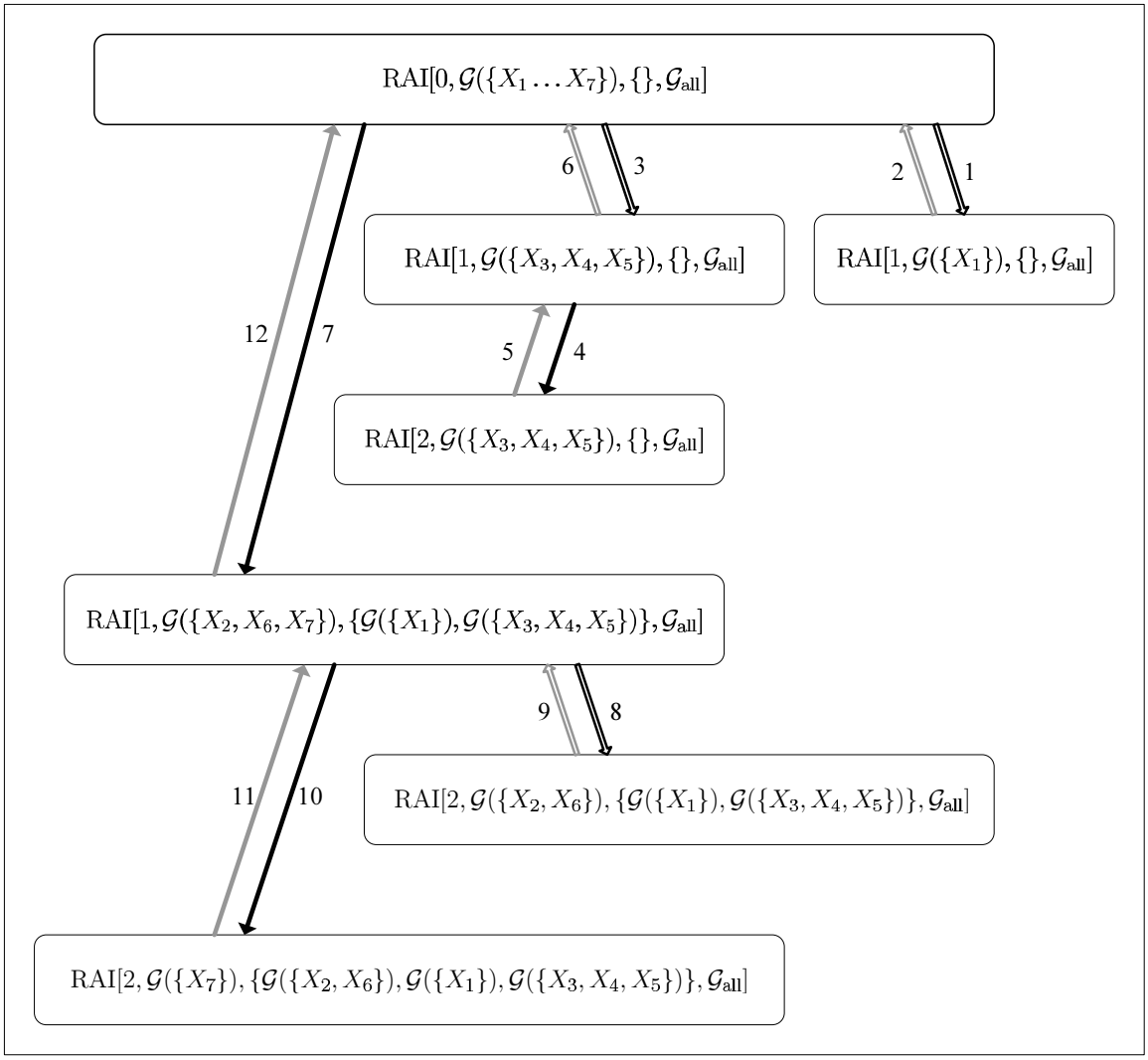


Figure 6: The execution order of the RAI algorithm for the example structure of Figure 5. Recursive calls of Stages C and D are marked with double and single arrows, respectively. The numbers annotating the arrows indicate the order of calls and returns of the algorithm.

empty and $\mathcal{G}_{\text{all}} = \mathcal{G}_{\text{start}}$, so Stage A is skipped. In Stage B1, any pair of nodes in $\mathcal{G}_{\text{start}}$ is CI tested given an empty condition set (i.e., checking marginal independence), which yields the removal of the edges between node X_1 and nodes X_3, X_4 and X_5 (Figure 5c). The edge directions inferred in Stage B2 are shown in Figure 5d. The nodes having the lowest topological order (X_2, X_6, X_7) are grouped into a descendant sub-structure $\mathcal{G}_{\text{D}}$ (Stage B3), while the remaining nodes form two unconnected ancestor sub-structures, $\mathcal{G}_{\text{A}_1}$ and $\mathcal{G}_{\text{A}_2}$ (Stage B4)(Figure 5e). Note that after decomposition, every edge between a node, X_i, in an ancestor sub-structure, and a node, X_j, in a descendant sub-structure is a directed edge $X_i \to X_j$. The set of all edges from an ancestor sub-structure to the descendant sub-structure is illustrated in Figure 5e by a wide arrow connecting the sub-structures. In Stage C, the algorithm is called recursively for each of the ancestor sub-structures with $n = 1$, $\mathcal{G}_{\text{start}} = \mathcal{G}_{\text{A}_i}$ $(i = 1,2)$ and $\mathcal{G}_{\text{ex}} = \emptyset$. Since sub-structure $\mathcal{G}_{\text{A}_1}$ contains a single node, the exit condition for this structure is satisfied. While calling $\mathcal{G}_{\text{start}} = \mathcal{G}_{\text{A}_2}$, Stage A is skipped, and in Stage B1 the algorithm identifies that $X_4 \perp\!\!\!\perp X_5 \,|\, X_3$, thus removing the edge $X_4 - X_5$. No orientations are identified (e.g., X_3 cannot be a collider, since it separated X_4 and X_5), so the three nodes have equal topological order and they are grouped to form a descendant sub-structure. The recursive call for this sub-structure with $n = 2$ is returned immediately, since the exit condition is satisfied (Figure 5f). Moving to Stage D, the RAI is called with $n = 1$, $\mathcal{G}_{\text{start}} = \mathcal{G}_{\text{D}}$ and $\mathcal{G}_{\text{ex}} = \{\mathcal{G}_{\text{A}_1}, \mathcal{G}_{\text{A}_2}\}$. Then, in Stage A1 relations $X_1 \perp\!\!\!\perp \{X_6, X_7\} \,|\, X_2$, $X_4 \perp\!\!\!\perp \{X_6, X_7\} \,|\, X_2$ and $\{X_3, X_5\} \perp\!\!\!\perp \{X_2, X_6, X_7\} \,|\, X_4$ are identified, and the corresponding edges are removed (Figure 5g). In Stage A2, X_6 and X_7 cannot collide at X_2 (since X_6 and X_7 are adjacent), and X_2 and X_6 (X_7) cannot collide at X_7 (X_6) (since X_2 and X_6 (X_7) are adjacent); hence, no additional V-structures are formed. Based on the inductive step and since X_1 is directed at X_2, X_2 should be directed at X_6 and at X_7. X_6 (X_7) cannot be directed at X_7 (X_6), because no new V-structures are allowed (Figure 5h). Stage B1 of the algorithm identifies the relation $X_2 \perp\!\!\!\perp X_7 \,|\, X_6$ and removes the edge $X_2 \to X_7$. In Stage B2, X_6 cannot be a collider of X_2 and X_7, since it has separated them. In the inductive step, X_6 is directed at X_7, $X_6 \to X_7$ (Figure 5i). In Stages B3 and B4, X_7 and $\{X_2, X_6\}$ are identified as a descendant sub-structure and an ancestor sub-structure, respectively. Further recursive calls (8 and 10 in Figure 6) are returned immediately, and the resulting partially directed structure (Figure 5i) represents a family of Markov equivalent structures (pattern) of the true structure (Figure 5a).

3.2 Minimality, Stability and Complexity

After describing the RAI algorithm (Section 3.1) and before proving its correctness (Section 3.3), we analyze in Section 3.2 three essential aspects of the algorithm—minimality, stability and complexity.

3.2.1 MINIMALITY

A structure recovered by the RAI algorithm in iteration m has a higher d-separation resolution and entails fewer dependences and thus is simpler and preferred[8] to a structure recovered in iteration $m-k$ where $0 < k \leq m$. By increasing the resolution, the RAI algorithm, similarly to the PC algorithm, moves from a complete undirected graph having maximal dependence relations between variables to structures having less (or equal) dependences than previous structures, ending in a structure having no edges between conditionally independent nodes, that is, a minimal structure.

8. We refer here to structures learned during algorithm execution and do not consider the empty graph that naturally has the lowest d-separation resolution (i.e., 0). This graph, having all nodes marginally independent of each other, will be found by the RAI algorithm immediately after the first iteration for graph resolution 0.

3.2.2 STABILITY

Similarly to Spirtes et al. (2000), we use the notion of stability informally to measure the number of errors in the output of a stage of the algorithm due to errors in the input to this stage. Similarly to the PC algorithm, the main sources of errors of the RAI algorithm are CI-testing and the identification of V-structures. Removal of an edge due to an erroneous CI test may lead to failure in correctly removing other edges, which are not in the true graph and also cause to orientation errors. Failure to remove an edge due to an erroneous CI test may prevent, or wrongly cause, orientation of edges. Missing or wrongly identifying a V-structure affect the orientation of other edges in the graph during the inductive stage and subsequent stages.

Many CI test errors (i.e., deciding that (in)dependence exists where it does not) in CB algorithms are the result of unnecessary large condition sets given a limited database size (Spirtes et al., 2000). Large condition sets are more likely to be inaccurate, since they are more likely to include unnecessary and erroneous nodes (erroneous due to errors in earlier stages of the algorithm). These sets may also cause poorer estimation of the criterion that measures dependence (e.g., CMI or χ^2) due to the curse-of-dimensionality, as typically there are only too few instances representing some of the combinations of node states. Either way, these condition sets are responsible for many wrong decisions about whether dependence between two nodes exists or not. Consequently, these errors cause structural inaccuracies and hence also poor inference ability.

Although CI-testing in the PC algorithm is more stable than V-structure identification (Spirtes et al., 2000), it is difficult to say whether this is also the case in the RAI algorithm. Being recursive, the RAI algorithm might be more unstable. However, CI test errors are practically less likely to occur, since by alternating between CI testing and edge direction the algorithm uses knowledge about parent-child relations before CI testing of higher orders. This knowledge permits avoiding some of the tests and decreases the size of conditions sets of some other tests (see Lemma 1). In addition, graph decomposition promotes decisions about well-founded orders of node presentation for subsequent CI tests, contrary to the common arbitrary order of presentation (see, e.g., the PC algorithm). Both mechanisms enhance stability and provide some means of error correction, as will be demonstrated shortly.

Let us now extensively describe examples that support our claim regarding the enhanced stability of the RAI algorithm. Suppose that following CI tests of some order both the PC and RAI algorithms identify a triplet of nodes in which two non-adjacent nodes, X and Y, are adjacent to a third node, Z, that is, $X - Z - Y$. In the immediate edge direction stage, the RAI algorithm identifies this triplet as a V-structure, $X \rightarrow Z \leftarrow Y$. Now, suppose that due to an unreliable CI test of a higher order the PC algorithm removes $X - Z$ and the RAI algorithm removes $X \rightarrow Z$. Eventually, both algorithms fail to identify the V-structure, but the RAI algorithm has an advantage over the PC algorithm in that the other arm of the V-structure is directed, $Z \leftarrow Y$. This contributes to the possibility to direct further edges during the inductive stage and subsequent recursive calls for the algorithm. The directed arm would also contribute to fewer CI tests and tests with smaller condition sets during CI testing with higher orders (e.g., if we later have to test independence between Y and another node, then we know that Z should not be included in the condition set, even though it is adjacent to Y). In addition, the direction of this edge also contributes to enhanced inference capability.

Now, suppose another example in which after removing all edges due to reliable CI tests using condition set sizes lower than or equal to n, the algorithm identifies the V-structure $X \rightarrow Z \leftarrow Y$ (Figure 7a). However, let assume that one of the V-structure arms, say $X \rightarrow Z$, is correctly removed

on a subsequent iteration using a larger condition set size (say $n+1$ without limiting the generality). We may be concerned that assuming a V-structure for the lower graph resolution, the RAI algorithm wrongly directs the second arm $Z - Y$ as $Z \leftarrow Y$. However, we demonstrate that the edge direction $Z \leftarrow Y$ remains valid even if there should be no edge $X - Z$ in the true graph. Suppose that $X \rightarrow Z$ was correctly removed conditioned on variable W, which is independent of Y given any condition set with a size smaller than or equal to n. Then, the possible underlying graphs are shown in Figures 7b-7d. The graph in Figure 7d is not possible, since it yields that X and Y are dependent given all condition sets of sizes smaller than or equal to n. In Figure 7b and Figure 7c, Z is a collider between W and Y, and thus the edge direction $Z \leftarrow Y$ remains valid. A different graph, $X \rightarrow W \leftarrow Z - Y$ (i.e., W is a collider), is not possible, since it means that $X \perp\!\!\!\perp Z \,|\, \boldsymbol{S}, |\boldsymbol{S}| \leq n, W \notin \boldsymbol{S}$ and then $X - Z$ should have been removed in a previous order (using condition set size of n or lower) and $X \rightarrow Z \leftarrow Y$ should not have been identified in the first place. Now, suppose that W and Y are dependant. In this case, the possible graphs are those shown in Figures 7e-7h. Similarly to the case in which W and Y are independent, W cannot be a collider of X and Z ($X \rightarrow W \leftarrow Z$) in this case as well. The graphs shown in Figures 7e-7g cannot be the underlying graphs since they entail dependency between X and Y given a condition set of size lower than or equal to n. The graph shown in Figure 7h exemplifies a V-structure $X \rightarrow W \leftarrow Y$. Since we assume that X and Z are independent given W (and thus $X - Z$ was removed), a V-structure $X \rightarrow W \leftarrow Z$ is not allowed. Since the edge $X \rightarrow W$ is already directed, the edge between W and Z must be directed as $W \rightarrow Z$. In this case, to avoid the cycle $Y \rightarrow W \rightarrow Z \rightarrow Y$, the edge between Y and Z must be directed as in the true graph, that is, $Y \rightarrow Z$.

Finally for the stability subsection, we note that the contribution of graph decomposition to structure learning using the RAI algorithm is threefold. First is the identification in early stages, using low-order, reliable CI tests, of the graph hierarchy, exemplifying the backbone of causal relations in the graph. For example, Figure 5e shows that learning our example graph (Figure 5a) from the complete graph (Figure 5b) demonstrates, immediately after the first iteration, that the graph is composed of three sub-structures—$\{X_1\}$, $\{X_2, X_6, X_7\}$ and $\{X_3, X_4, X_5\}$, where $\{X_1\} \rightarrow \{X_2, X_6, X_7\}$ and $\{X_3, X_4, X_5\} \rightarrow \{X_2, X_6, X_7\}$. This rough (low-resolution) partition of the graph is helpful in visualizing the problem and representing the current knowledge from the outset and along the learning. The second contribution of graph decomposition is the possibility to implement learning using a parallel processor for each sub-structure independently. This advantage may be further extended in the recursive calls for the algorithm.

Third is the contribution of graph decomposition to improved performance. Aiming at a low number of CI tests, decomposition provides a sound guideline for deciding on an educated order in which the edges should be CI tested. Based on this order, some tests can be considered redundant and thus be avoided. Several methods for selecting the right order for the PC algorithm were presented in Spirtes et al. (2000), but these methods are heuristic. Decomposition into ancestor and descendent sub-structures is followed by three levels of learning (Figure 4), that is, removing and directing edges 1) of ancestor sub-structures, 2) between ancestor and descendent sub-structures, and 3) of the descendent sub-structure. The second level has the greatest influence on further learning. The removal of edges between ancestor and descendent sub-structures and the sequential direction of edges in the descendant sub-structure assure that, first, fewer potential parents are considered, while learning the descendent sub-structure and second, more edges can be directed in this latter sub-structure. Moreover, these directed edges and the derived parent-child relations prevent an arbitrary selection order of nodes for CI testing and thereby enable employing smaller and more

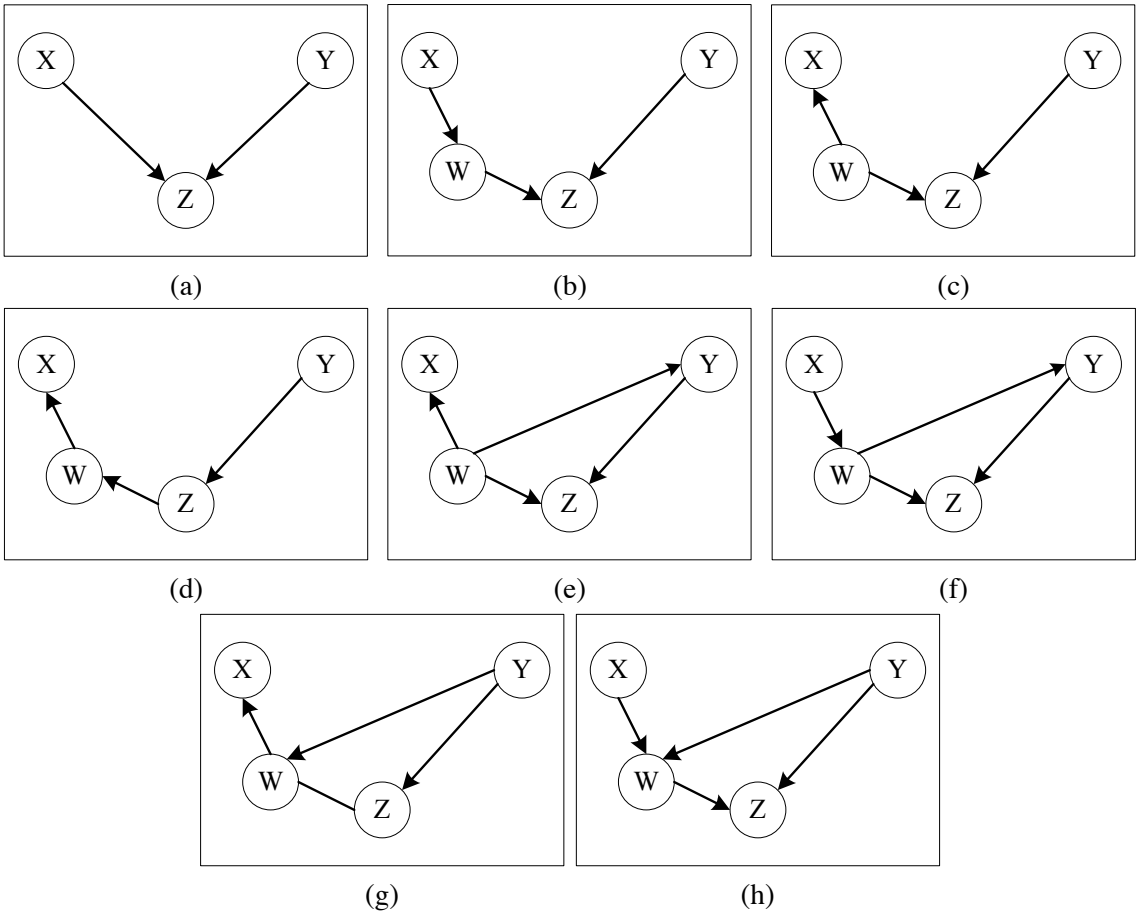


Figure 7: Graphs used to exemplify the stability of the RAI algorithm (see text).

accurate condition sets. Take, for example, CI testing for the redundant edge between X_2 and X_7 in our example graph (Figure 5i) if the RAI algorithm did not use decomposition. Graph decomposition for $n = 0$ (Figure 5e) enables the identification of two ancestor sub-structures, $\mathcal{G}_{\mathrm{A}_1}$ and $\mathcal{G}_{\mathrm{A}_2}$, as well as a descendent sub-structure $\mathcal{G}_{\mathrm{D}}$ that are each learned recursively. During Stage D (Figure 4) and while thinning the links between the ancestor sub-structures and $\mathcal{G}_{\mathrm{D}}$ (in Stage A1 of the recursion for $n = 1$), we identify the relations $X_1 \perp\!\!\!\perp \{X_6, X_7\} \,|\, X_2$, $X_4 \perp\!\!\!\perp \{X_6, X_7\} \,|\, X_2$ and $\{X_3, X_5\} \perp\!\!\!\perp \{X_2, X_6, X_7\} \,|\, X_4$ and remove the 10 corresponding edges (Figure 5g). The decision to test and remove these edges first was enabled by the decomposition of the graph to $\mathcal{G}_{\mathrm{A}_1}$, $\mathcal{G}_{\mathrm{A}_2}$ and $\mathcal{G}_{\mathrm{D}}$. In Stage A2 (Figure 5h), we direct the edge $X_2 \to X_6$ (as $X_1 \perp\!\!\!\perp X_6 \,|\, X_2$ and thus X_2 cannot be a collider between X_1 and X_6) and edge $X_2 \to X_7$ (as $X_1 \perp\!\!\!\perp X_7 \,|\, X_2$ and thus X_2 cannot be a collider between X_1 and X_7), and in Stage B (Figure 5i) we direct the edge $X_6 \to X_7$. The direction of these edges could not be assured without removing first the above edges, since the (redundant) edges pointing onto X_6 and X_7 would have allowed wrong edge direction, that is, $X_6 \to X_2$ and $X_7 \to X_2$. If we had been using the RAI algorithm with no decomposition (Figure 5d) (or the PC algorithm) and had decided to check the independence between X_2 and X_7, first, we would have had to consider condition sets containing the nodes X_1, X_3, X_4, X_5 or X_6 (up to 10 CI tests whether we start from

X_2 or X_7). Instead, we perform in Stage B1 only one test, $X_2 \perp\!\!\!\perp X_7 \,|\, X_6$. These benefits are the result of graph decomposition.

3.2.3 COMPLEXITY

CI tests are the major contributors to the (run-time) complexity of CB algorithms (Cheng and Greiner, 1999). In the worst case, the RAI algorithm will neither direct any edges nor decompose the structure and will thus identify the entire structure as a descendant sub-structure, calling Stages D and B1 iteratively while skipping all other stages. Then, the execution of the algorithm will be similar to that of the PC algorithm, and thus the complexity will be bounded by that of the PC algorithm. Given the maximal number of possible parents k and the number of nodes n, the number of CI tests is bounded by (Spirtes et al., 2000)

$$2\binom{n}{2}\cdot\sum_{i=0}^{k}\binom{n-1}{i}\leq\frac{n^2(n-1)^{k-1}}{(k-1)!},$$

which leads to complexity of $O(n^k)$.

This bound is loose even in the worst case (Spirtes et al., 2000) especially in real-world applications requiring graphs having V-structures. This means that in most cases some edges are directed and the structure is decomposed; hence, the number of CI tests is much smaller than that of the worst case. For example, by decomposing our example graph (Figure 5) into descendent and ancestor sub-structures in the first application of Stage B4 (Figure 5e), we avoid checking $X_6 \perp\!\!\!\perp X_7 \,|\, \{X_1, X_3, X_4, X_5\}$. This is because $\{X_1, X_3, X_4, X_5\}$ are neither X_6's nor X_7's parents and thus are not included in the (autonomous) descendent sub-structure. By checking only $X_6 \perp\!\!\!\perp X_7 \,|\, \{X_2\}$, the RAI algorithm saves CI tests that are performed by the PC algorithm. We will further elaborate on the RAI algorithm complexity in our forthcoming study.

3.3 Proof of Correctness

We prove the correctness of the RAI algorithm using Proposition 2. We show that only conditional independences (of all orders) entailed by the true underlying graph are identified by the RAI algorithm and that all V-structures are correctly identified. We then note on the correctness of edge direction.

Proposition 2: If the input data to the RAI algorithm are faithful to a DAG, $\mathcal{G}_{\text{true}}$, having any d-separation resolution, then the algorithm yields the correct pattern for $\mathcal{G}_{\text{true}}$.

Proof: We use mathematical induction to prove the proposition, where in each induction step, m, we prove that the RAI algorithm finds (a) all conditional independences of order m and lower, (b) no false conditional independences, (c) only correct V-structures and (d) all V-structures, that is, no V-structures are missing.

Base step ($m = 0$): If the input data to the RAI algorithm was generated from a distribution faithful to a DAG, $\mathcal{G}_{\text{true}}$, having d-separation resolution 0, then the algorithm yields the correct pattern for $\mathcal{G}_{\text{true}}$.

Given that the true underlying DAG has a d-separation resolution of 0, the data entail only marginal independences. In the beginning of learning, $\mathcal{G}_{\text{start}}$ is a complete graph and $m = 0$. Since

there are no exogenous causes, Stage A is skipped. In Stage B, the algorithm tests for independence between every pair of nodes with an empty condition set, that is, $X \perp\!\!\!\perp Y \,|\, \emptyset$ (marginal independence), removes the redundant edges and directs the remaining edges as possible. In the resulting structure, all the edges between independent nodes have been removed and no false conditional independences are entailed. Thus, all the identified V-structures are correct, as discussed in Section 3.2.2 on stability, and there are no missing V-structures, since the RAI algorithm has tested independence for all pair of nodes (edges). At the end of Stage B2 (edge direction), the resulting structure and $\mathcal{G}_{\text{true}}$ have the same set of V-structures and the same set of edges. Thus, the correct pattern for $\mathcal{G}_{\text{true}}$ is identified. Since the data entail only independences of zero order, further recursive calls with $m \geq 1$ will not find independences with condition sets of size m, and thus no edges will be removed, leaving the graph unchanged.

Inductive step $(m+1)$: Suppose that at induction step m, the RAI algorithm discovers all conditional independences of order m and lower, no false conditional independences are entailed, all V-structures are correct, and no V-structures are missing. Then, if the input data to the RAI algorithm was generated from a distribution faithful to a DAG, $\mathcal{G}_{\text{true}}$, having d-separation resolution $m+1$, then the RAI algorithm would yield the correct pattern for that graph.

In step m, the RAI algorithm discovers all conditional independences of order m and lower. Given input data faithful to a DAG, $\mathcal{G}_{\text{true}}$, having d-separation resolution $m+1$, there exists at least one pair of nodes, say $\{X,Y\}$, in the true graph, that has a d-separation resolution of $m+1$.[9] Since the RAI, by the recursive call $m+1$ (i.e., calling RAI$[m+1, \mathcal{G}_{\text{start}}, \mathcal{G}_{\text{ex}}, \mathcal{G}_{\text{all}}]$), has identified only conditional independences of order m and lower, an edge, $E_{XY} = (X - Y)$, exists in the input graph, $\mathcal{G}_{\text{start}}$. The smallest condition set required to identify the independence between X and Y is $\boldsymbol{S}_{XY}$ ($X \perp\!\!\!\perp Y \,|\, \boldsymbol{S}_{XY}$), such that $|\boldsymbol{S}_{XY}| \geq m+1$. Thus, $|\boldsymbol{Pa}_p(X) \backslash Y| \geq m+1$ or $|\boldsymbol{Pa}_p(Y) \backslash X| \geq m+1$, meaning that either node X or node Y has at least $m+2$ potential parents. Such an edge exists in at least one of the autonomous sub-structures decomposed from the graph yielded at the end of iteration m. When calling, in Stage C or Stage D, the algorithm recursively for this sub-structure with $m' = m+1$, the exit condition is not satisfied because either node X or node Y has at least $m'+1$ parents. Since Step m assured that the sub-structure is autonomous, it contains all the necessary node parents. Note that decomposition into ancestor, $\mathcal{G}_{\text{A}}$, and descendant, $\mathcal{G}_{\text{D}}$, sub-structures occurs after identification of all nodes having the lowest topological order, such that every edge from a node X in $\mathcal{G}_{\text{A}}$ to a node Y in $\mathcal{G}_{\text{D}}$ is directed, $X \rightarrow Y$. In the case that the sub-structure is an ancestor sub-structure, $\boldsymbol{S}_{XY}$ contains nodes of the sub-structure and its exogenous causes. In the case that the sub-structure is a descendant sub-structure, $\boldsymbol{S}_{XY}$ contains nodes from the ancestor sub-structures and the descendant sub-structure. Therefore, based on Proposition 1, the RAI algorithm tests all edges using condition sets of sizes m' and removes E_{XY} (and all similar edges) in either Stage A or Stage B, yielding a structure with d-separation resolution of m' and thereby yields the correct pattern for the true underlying graph of d-separation resolution $m+1$.

Spirtes (2001)—when introducing the anytime fast casual inference (AFCI) algorithm—proved the correctness of edge direction of AFCI. The AFCI algorithm can be interrupted at any stage (resolution), and the resultant graph at this stage is correct with probability one in the large sample

9. If the d-separation resolution of $\{X,Y\}$ is $m' > m+1$, then the RAI algorithm will not modify the graph until step m'.

limit, although possibly less informative[10] than if had been allowed to continue uninterrupted.[11] Recall that interrupting learning means that we avoid CI tests of higher orders. This renders the resultant graph more reliable. We use this proof here for proving the correctness of edge direction in the RAI algorithm. Completing CI testing with a specific graph resolution n in the RAI algorithm and interrupting the AFCI at any stage of CI testing are analogous. Furthermore, Spirtes (2001) proves that interrupting the algorithm at any stage is also possible during edge direction, that is, once an edge is directed, the algorithm never changes that direction. In Section 3.2.2, we showed that even if a directed edge of a V-structure is removed, the direction of the remaining edge is still correct. Since directing edges by the AFCI algorithm after interruption yields a correct (although less informative) graph (Spirtes, 2001), also the direction of edges by the RAI algorithm yields a correct graph. Having (real) parents in a condition set used for CI testing, instead of potential parents, which are the result of edge direction for resolutions lower than n, is a virtue, as was confirmed in Section 3.1. All that is required that all parents, either real or potential, be included within the corresponding condition set, and this is indeed guaranteed by the autonomy of each substructure, as was proved above. ■

4. Experiments and Results

We compare the RAI algorithm with other state-of-the-art algorithms with respect to structural correctness, computational complexity, run-time and classification accuracy when the learned structure is used in classification. The algorithms learned structures from databases representing synthetic problems, real decision support systems and natural classification problems. We present the experimental evaluation in four sections. In Section 4.1, the complexity of the RAI algorithm is measured by the number of CI tests required for learning synthetically generated structures in comparison to the complexity of the PC algorithm (Spirtes et al., 2000).

The order of presentation of nodes is not an input to the PC algorithm. Nevertheless, CI testing of orders higher than 0, and therefore also edge directing, which depends on CI testing, may be sensitive to that order. This may cause learning different graphs whenever the order is changed. Dash and Druzdzel (1999) turned this vice of the PC algorithm into a virtue by employing the partially directed graphs formed by using different orderings for the PC algorithm as the search space from which the structure having the highest value of the K2 metric (Cooper and Herskovits, 1992) is selected. For the RAI algorithm, sensitivity to the order of presentation of nodes is expected to be reduced compared to the PC algorithm, since the RAI algorithm, due to edge direction and graph decomposition, decides on the order of performing most of the CI tests and does not use an arbitrary order (Section 3.2.2). Nevertheless, to account for the possible sensitivity of the RAI and PC algorithms to this order, we preliminarily employed 100 different permutations[12] of the order for each of ten Alarm network (Beinlich et al., 1989) databases. Since the results of these experiments

10. Less informative in the sense that it answers "can't tell" for a larger number of questions; that is, identifying, for example, "$\circ$" edge endpoint (placing no restriction on the relation between the pair of nodes making the edge) instead of "$\rightarrow$" endpoint.
11. The AFCI algorithm is also correct if hidden and selection variables exist. A selection variable models the possibility of an observable variable having some missing data. We focus here on the case where neither hidden nor selection variables exist.
12. Dash and Druzdzel (1999) examined the relationships between the number of order permutations and the numbers of variables and instances. We fixed the number of order permutations at 100.

had showed that the difference in performance for different permutations is slight, we further limited the experiments with the PC and RAI algorithms to a single permutation.

In Section 4.2, we present our methodology of selecting a threshold for RAI CI testing. We propose selecting a threshold for which the learned structure has a maximum of a likelihood-based score value.

In Section 4.3, we use the Alarm network (Beinlich et al., 1989), which is a widely accepted benchmark for structure learning, to evaluate the structural correctness of graphs learned by the RAI algorithm. The correctness of the structure recovered by RAI is compared to those of structures learned using other algorithms—PC, TPDA (Cheng et al., 1997), GES (Chickering, 2002; Meek, 1997), SC (Friedman et al., 1999) and MMHC (Tsamardinos et al., 2006a). The PC and TPDA algorithms are the most popular CB algorithms (Cheng et al., 2002; Kennett et al., 2001; Marengoni et al., 1999; Spirtes et al., 2000); GES and SC are state-of-the-art S&S algorithms (Tsamardinos et al., 2006a); and MMHC is a hybrid algorithm that has recently been developed and showed superiority, with respect to different criteria, over all the (non-RAI) algorithms examined here (Tsamardinos et al., 2006a). In addition to correctness, the complexity of the RAI algorithm, as measured through the enumeration of CI tests and log operations, is compared to those of the other CB algorithms (PC and TPDA) for the Alarm network.

In Section 4.4, we extend the examination of RAI in structure learning to known networks other than the Alarm. Although the Alarm is a popular benchmark network, many algorithms perform well for this network. Hence, it is important to examine RAI performance on other networks for which the true graph is known. In the comparison of RAI to other algorithms, we included all the algorithms of Section 4.3, as well as the Optimal Reinsertion (OR) (Moore and Wong, 2003) algorithm and a greedy hill-climbing search algorithm with a Tabu list (GS) (Friedman et al., 1999). We compared algorithm performances with respect to structural correctness, run-time, number of statistical calls and the combination of correctness and run-time.

In Section 4.5, the complexity and run-time of the RAI algorithm are compared to those of the PC algorithm using nineteen natural databases. In addition, the classification accuracy of the RAI algorithm for these databases is compared to those of the PC, TPDA, GES, MMHC, SC and naive Bayesian classifier (NBC) algorithms. No structure learning is required for NBC and all the domain variables are used. This classifier is included in the study as a reference to a simple, yet accurate, classifier. Because we are interested in this section in classification, and a likelihood-based score does not reflect the importance of the class variable in structures used for classification (Friedman et al., 1997; Kontkanen et al., 1999; Grossman and Domingos, 2004; Yang and Chang, 2002), we prefer here the classification accuracy score in evaluating structure performance.

In the implementations of all sections, except Section 4.4, we were aided by the Bayes net toolbox (BNT) (Murphy, 2001), BNT structure learning package (Leray and François, 2004) and PowerConstructor software (Cheng, 1998) and evaluated all algorithms ourselves. In Section 4.4, we downloaded and used the results reported in Tsamardinos et al. (2006a) for the non-RAI algorithms and used the Causal Explorer algorithm library (Aliferis et al., 2003) (http://www.dsl-lab.org/causal_explorer/index.html). The Causal Explorer algorithm library makes use of methods and values of parameters for each algorithm as suggested by the authors of each algorithm (Tsamardinos et al., 2006a). For example, BDeu score (Heckerman et al., 1995) with equivalent sample size 10 for GS, GES, OR and MMHC; χ^2 p-values at the standard 5% for the MMHC's and PC's statistical thresholds; threshold of 1% for the TPDA mutual information test; the Bayesian scoring heuristic, equivalent sample size of 10 and maximum allowed sizes for the candidate parent

set of 5 and 10 for SC; and maximum number of parents allowed of 5, 10 and 20 and maximum allowed run time, which is one and two times the time used by MMHC on the corresponding data set, for OR. The only parameter that requires optimization in the RAI algorithm (similar to the other CB algorithms - PC and TPDA) is the CI testing threshold. We use no prior knowledge to find this threshold but a training set for each database (see Section 4.2 for details). Note, however that we do not account for the time required for selecting the threshold when reporting the execution time.

4.1 Experimentation with Synthetic Data

The complexity of the RAI algorithm was evaluated in comparison to that of the PC algorithm by the number of CI tests required to learn synthetically generated structures. Since the true graph is known for these structures, we could assume that all CI tests were correct and compare the numbers of CI tests required by the algorithms to learn the true independence relationships. In one experiment, all 29,281 possible structures having 5 nodes were learned using the PC and RAI algorithms. The average number of CI tests employed by each algorithm is shown in Figure 8a for increasing orders (condition set sizes). Figure 8b depicts the average percentages of CI tests saved by the RAI algorithm compared to the PC algorithm for increasing orders. These percentages were calculated for each graph independently and then averaged. It is seen that the advantage of the RAI algorithm over the PC algorithm is more prominent for high orders.

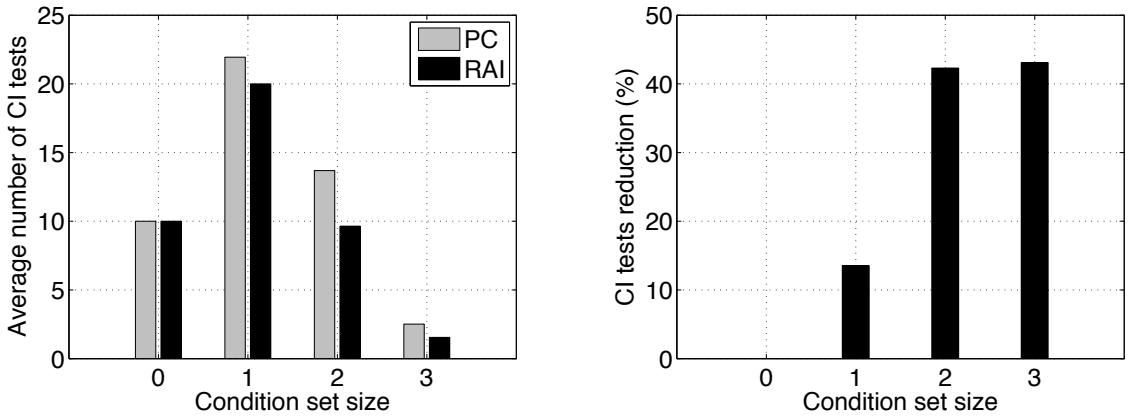


Figure 8: Measured for increasing orders, the (a) average number of CI tests required by the RAI and PC algorithms for learning all possible structures having five nodes and (b) average over all structures of the reduction percentage in CI tests achieved by the RAI algorithm compared to the PC algorithm.

In another experiment, we learned graphs of sizes (numbers of nodes) between 6 and 15. We selected from a large number of randomly generated graphs 3,000 graphs that were restricted by a maximal fan-in value of 3; that is, every node in such a graph has 3 parents at most and at least one node in the graph has 3 parents. This renders a practical learning task. Thus, the structures can theoretically be learned by employing CI tests of order 3 and below and should not use tests of orders higher than 3. In such a case, the most demanding test, having the highest impact on

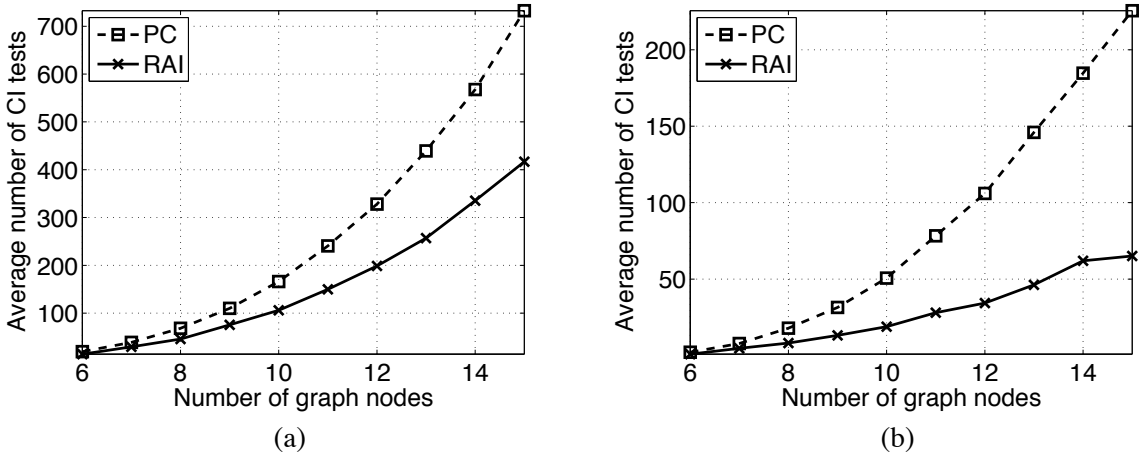


Figure 9: Average number of CI tests required by the PC and RAI algorithms for increasing graph sizes and orders of (a) 3 and (b) 4.

computational time, is of order 3. Figure 9a shows the average numbers of CI tests performed for this order by the PC and RAI algorithms for graphs with increasing sizes. Moreover, because the maximal fan-in is 3, all CI tests of order 4 are a priori redundant, so we can further check how well each algorithm avoids these unnecessary tests. Figure 9b depicts the average numbers of CI tests performed by the two algorithms for order 4 and graphs with increasing sizes. Both Figure 9a and Figure 9b show that the number of CI tests employed by the RAI algorithm increases more slowly with the graph size compared to that of the PC algorithm and that this advantage is much more significant for the redundant (and more costly) CI tests of order 4.

We further expanded the examination of the algorithms in CI testing for different graph sizes and CI test orders. Figure 10 shows the average number and percentage of CI tests saved using the RAI algorithm compared to the PC algorithm for different condition set sizes and graph sizes. The number of CI tests having an empty condition set employed by each of the algorithms is equal and is therefore omitted from the comparison. The figure shows that the percentage of CI tests saved using the RAI algorithm increases with both graph and condition set sizes. For example, the saving in CI tests when using the RAI algorithm instead of the PC algorithm for learning a graph having 15 nodes and using condition sets of size 4 is above 70% (Figure 10b). In Section 4.4, we will demonstrate the RAI quality of requiring relatively fewer tests of high orders than of low orders for graphs of larger sizes for real, rather than synthetic, data.

4.2 Selecting the Threshold for RAI CI Testing

CI testing for the RAI algorithm can be based on the χ^2 test as for the PC algorithm or the conditional mutual information (CMI) as for the TPDA algorithm. The CMI between nodes X and Y conditioned on a set of nodes $\boldsymbol{Z}$ (i.e., the condition set), is:

$$\mathrm{CMI}(X,Y|\boldsymbol{Z}) = \sum_{i=1}^{N_X}\sum_{j=1}^{N_Y}\sum_{k=1}^{N_Z}\left[P(x_i,y_j,\boldsymbol{z}_k)\cdot\log\frac{P(x_i,y_j|\boldsymbol{z}_k)}{P(x_i|\boldsymbol{z}_k)\cdot P(y_j|\boldsymbol{z}_k)}\right], \tag{2}$$

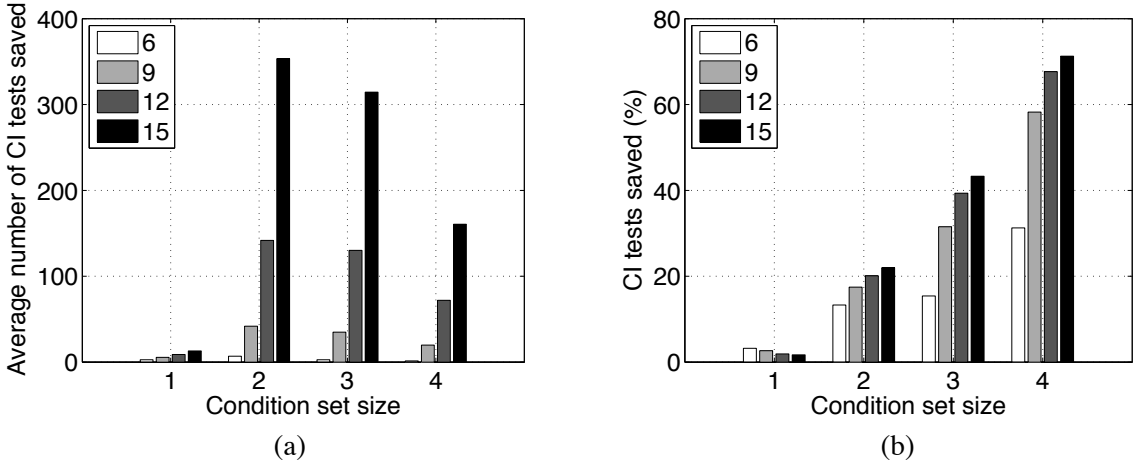


Figure 10: (a) Average number and (b) percentage of CI tests saved by using the RAI algorithm compared to the PC algorithm for graph sizes of 6, 9, 12 or 15 (gray shades) and orders between 1 and 4.

where x_i and y_j represent, respectively, states of X and Y, z_k represents a combination of states of all variables in $\boldsymbol{Z}$, and N_X, N_Y and N_Z are the numbers of states of X, Y and $\boldsymbol{Z}$, respectively.

In both CI testing methods, the value of interest (either χ^2 or CMI) is compared to a threshold. For example, CMI values that are higher or lower than the threshold indicate, respectively, conditional dependence or independence between X and Y given $\boldsymbol{Z}$. However, the optimal threshold is unknown beforehand. Moreover, the optimal threshold is problem and data-driven, that is, it depends, on the one hand, on the database and its size and, on the other hand, on the variables and the numbers of their states. Thus, it is not possible to set a "default" threshold value that will accurately determine conditional (in)dependence while using any database or problem.

To find an optimal threshold for a database, we propose to score structures learned using different thresholds by a likelihood-based criterion evaluated using the training (actually validation) set and to select the threshold leading to the structure achieving the highest score. Such a score may be BDeu (Heckerman et al., 1995), although other scores (Heckerman et al., 1995) may also be appropriate. Note that BDeu scores equally statistically indistinguishable structures. Figure 11 shows BDeu values for structures learned by RAI for the Alarm network using different CMI threshold values. The maximum BDeu value was achieved at a threshold value of 4e-3 that was selected as the threshold for RAI CI testing for the Alarm network.

To assess the threshold selected using the suggested method, we employed the Alarm network and computed the errors between structures learned using different thresholds and the pattern that corresponds to the true known graph. Following Spirtes et al. (2000) and Tsamardinos et al. (2006a), we define five types of structural errors to evaluate structural correctness. An extra edge (commission; EE) error is due to an edge learned by the algorithm although it does not exist in the true graph. A missing edge (omission; ME) error is due to an edge missed by the algorithm although exists in the true graph. An extra direction (ED) error is due to edge direction that appears in the learned graph but not in the true graph, whereas a missing direction (MD) error is due to edge direction that

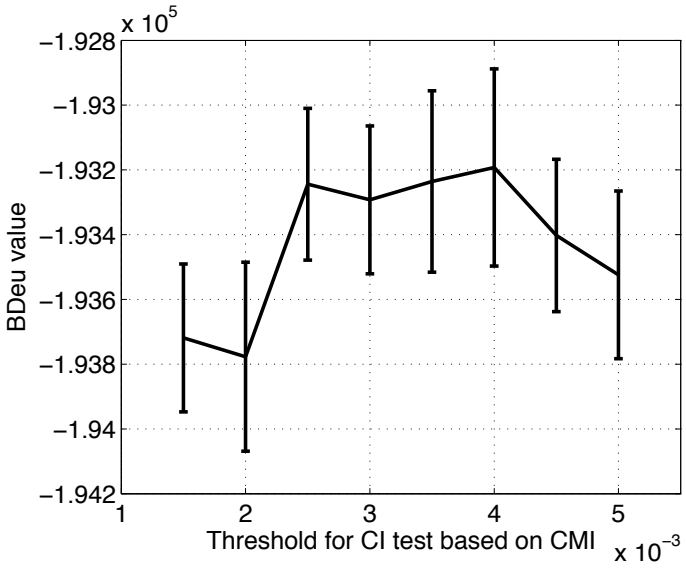


Figure 11: BDeu values averaged over ten validation sets consisting of 10,000 samples each drawn from the Alarm network for increasing CMI thresholds used in CI testing for the RAI algorithm.

appears in the true graph but not in the learned graph. Finally, a reversed direction (RD) error is due to edge direction in the learned graph that is opposite to the edge direction in the true graph.

Figure 12a shows the sensitivity of the five structural errors to the CMI threshold. Each point on the graph is the average error over ten validation databases containing 10,000 randomly sampled instances each. Figure 12a demonstrates that the MD, RD and ED errors are relatively constant in the examined range of thresholds and the ME error increases monotonically. The EE error is the highest error among the five error types, and it has a minimum at a threshold value of 3e-3.

In Figure 12b, we cast the three directional errors using the total directional error (DE), DE = ED + MD + RD, and plot this error together with the ME and EE errors. The impact of each error for increasing thresholds is now clearer; the contribution of the DE error is almost constant, that of the ME error increases with the threshold but is less than DE, and that of the EE error dominants for every threshold.

Tsamardinos et al. (2006a) suggested assessing the quality of a learned structure using the structural Hamming distance (SHD) metric, which is the sum of the five above errors. We plot in Figure 12c this error for the experiment with the Alarm network. Comparison of the threshold responsible for the minimum of the SHD error (2.5e-3) to that selected according to BDeu (4e-3 in Figure 11) shows only a small difference, especially as the maximum values of BDeu are obtained between thresholds of 2.5e-3 and 4e-3. This result motivates using the BDeu score, as measured on a validation set, as a criterion for finding good thresholds for RAI CI testing. Thresholds that are smaller than this range lead to too many pairs of variables that are wrongly identified as dependent and thus the edges between them are not removed, contributing to high EE errors (see, for example, Figure 12b). In addition, for thresholds higher than 3e-3, more edges are wrongly removed, contributing to high ME errors.

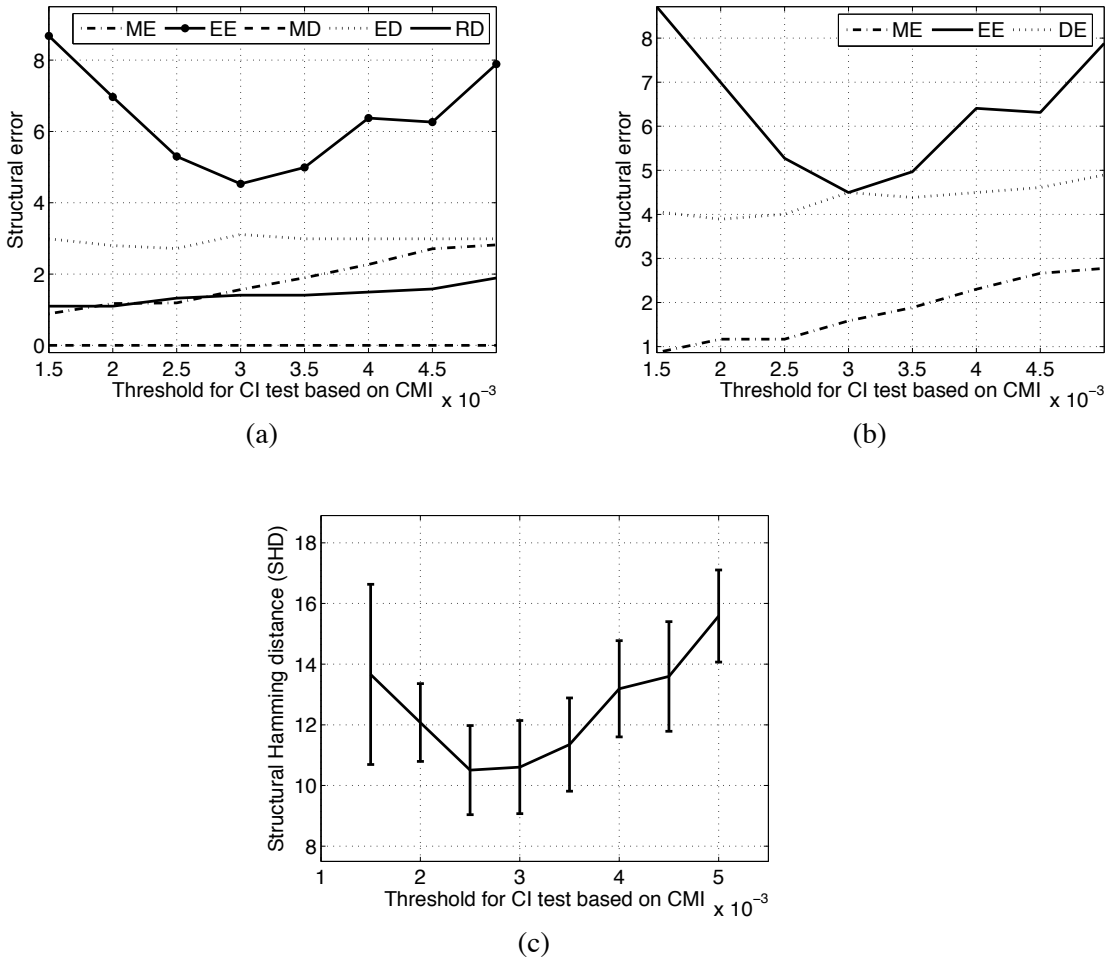


Figure 12: Structural errors of the RAI algorithm learning the Alarm network for different CMI thresholds as averaged over ten validation sets of 10,000 samples each. (a) Five types (ME, EE, MD, ED and RD) of structural errors, (b) EE, ME and DE errors, and (c) SHD error (mean and std).

4.3 Learning the Alarm Network

For evaluating the correctness of learned BN structures, we used the Alarm network, which is widely accepted as a benchmark for structure learning algorithms, since the true graph for this problem is known. The RAI algorithm was compared to the PC, TPDA, GES, SC and MMHC algorithms using ten databases containing 10,000 random instances each sampled from the network.

Structural correctness can be measured using different scores. However, some of the scores suggested in the literature are not always accurate or related to the true structure. For example, Tsamardinos et al. (2006a), who examined the BDeu score (Heckerman et al., 1995) and KL divergence (Kullback and Leibler, 1951) in evaluating learned networks, noted that it is not known in practice to what degree the assumptions (e.g., a Dirichlet distribution of the hyperparameters) in the

	Extra Direction (ED)	Missing Direction (MD)	Reversed Direction (RD)	Directional Error (DE)	Extra Edge (EE)	Missing Edge (ME)	SHD
SC	1	9.5	4.6	15.1	4.7	4.5	24.3
MMHC	0.8	3.3	5.7	9.8	2.6	**0.7**	13.1
GES	0.1	0.6	1.2	1.9	2.7	0.8	5.4
TPDA	**0**	4.2	**0**	4.2	2.4	2.9	9.5
PC	**0**	**0**	0.8	0.8	2.5	1.0	4.3
RAI	**0**	**0**	0.3	**0.3**	**1.8**	1.4	**3.5**

Table 1: Structural errors of several algorithms as averaged over 10 databases each containing 10,000 randomly generated instances of the Alarm network. The total directional error is the sum of three different directional errors, DE=ED+MD+RD, and the SHD error is DE+EE+ME. Bold font emphasizes the smallest error over all algorithms for each type of structural error.

basis of the BDeu score hold. Moreover, usually such a score is used in both learning and evaluation of a structure; hence the score favors algorithms that use it in learning. Tsamardinos et al. (2006a) also mentioned that both scores do not rely on the true structure. Thus, they suggested the SHD metric, which is directly related to structural correctness, since it is the sum of the five errors of Section 4.2. Nevertheless, since SHD can be measured only when the true graph is known, scores such as BDeu and KL divergence are of great value in practical situations, for example, in classification problems like those examined in Section 4.5 in which the true graph is not known. These scores are also beneficial in the determination of algorithm parameters. For example, in Section 4.2 we measured BDeu scores of structures learned using different thresholds in order to select a good threshold for RAI CI testing.

Although SHD sums all five structural errors, we were first interested in examining the contribution of each individual error to the total error. Table 1 summarizes the five structural errors for each algorithm as averaged over 10 databases of 10,000 instances each sampled from the Alarm network. These databases are different from those validation databases used for threshold setting. The table also shows the total directional error, DE, which is the sum of the three directional errors. Table 1 demonstrates that the lowest EE and DE errors are achieved by the RAI algorithm and the lowest ME error is accomplished by the MMHC algorithm. Computing SHD shows the advantage of the RAI (3.5) algorithm over the PC (4.3), TPDA (9.5), GES (5.4), MMHC (13.1) and the SC (24.3) algorithms. Further, we propose such a table as Table 1 as a useful tool for the identification of the sources of structural errors of a given structure learning algorithm.

Note that the SHD error weighs each of the five error types equally. We believe that a score that weighs the five types based on their relative significance to structure learning will be a more accurate method to evaluate structural correctness; however, deriving such a score is a topic for future research.

Complexity was evaluated for each of the CB algorithms by measuring the number of CI tests employed for each order (condition set size) and the total number of log operations. The latter criterion is proportional to the total number of multiplications, divisions and logarithm evaluations that is required for calculating the CMI (Equation 2) during CI testing. Figure 13 depicts the average

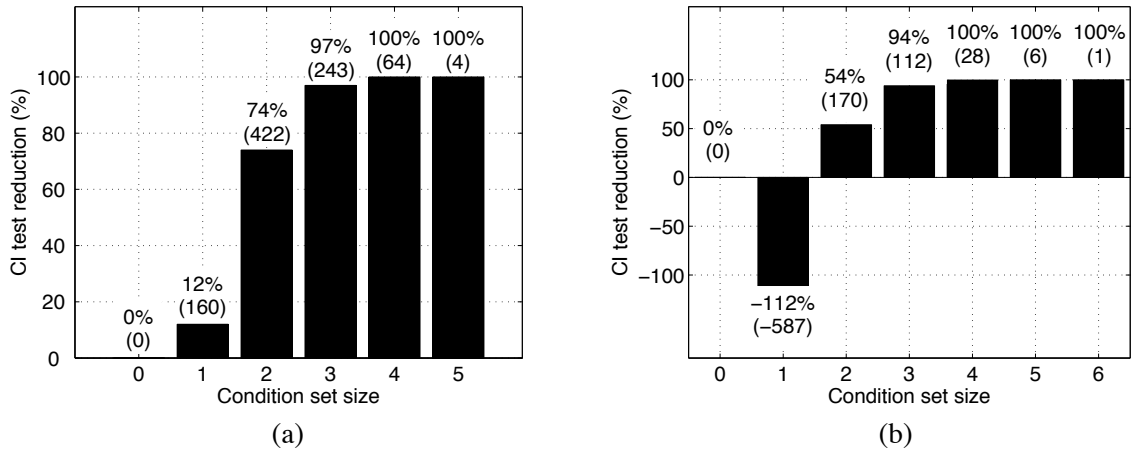


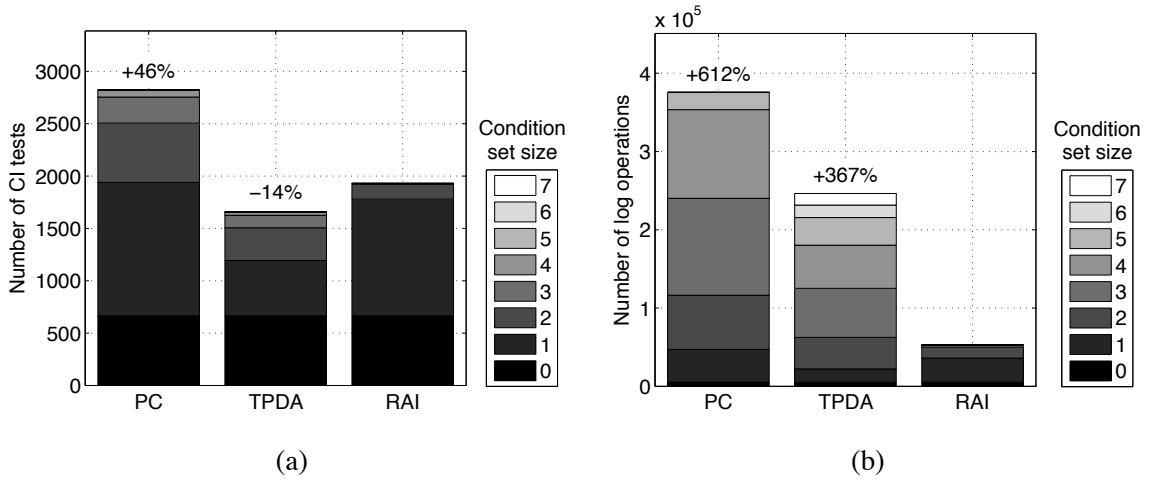
Figure 13: Average percentage (number) of CI tests reduced by using RAI compared to using (a) PC and (b) TPDA, as a function of the condition set size when learning the Alarm network.

percentage (and number) of CI tests reduced by using the RAI algorithm compared to using the PC or TPDA algorithms for increasing sizes of the condition sets. The RAI algorithm reduces the number of CI tests of orders 1 and above required by the PC algorithm and those of orders 2 and above required by the TPDA algorithm. Moreover, the RAI algorithm completely avoids the use of CI tests of orders 4 and above and almost completely avoids CI tests of order 3 compared to both the PC and TPDA algorithms. However, the RAI algorithm performs more CI tests of order 1 than the TPDA algorithm.

Figure 14 summarizes the total numbers of CI tests and log operations over different condition set sizes required by each algorithm. The RAI algorithm requires 46% less CI tests than the PC algorithm and 14% more CI tests (of order 1) than the TPDA algorithm. However, the RAI algorithm significantly reduces the number of log operations required by the other two algorithms. The PC or TPDA algorithms require, respectively, an additional 612% or 367% of the number of log operations required by the RAI algorithm. The reason for this substantial advantage of the RAI algorithm over both the PC and TPDA algorithms is the saving in CI tests of high orders (see Figure 13). These tests make use of large condition sets and thus are very expensive computationally.

4.4 Learning Known Networks

In addition to the state-of-art algorithms that were compared in Section 4.3, we include in this section the OR and GS algorithms. We compare the performance of the RAI algorithm to these algorithms by learning the structures of known networks employed in real decision support systems from a wide range of applications. We use known networks described in Tsamardinos et al. (2006a), which include the Alarm (Beinlich et al., 1989), Barley (Kristensen and Rasmussen, 2002), Child (Cowell et al., 1999), Hailfinder (Jensen and Jensen, 1996), Insurance (Binder et al., 1997), Mildew (Jensen and Jensen, 1996) and Munin (Andreassen et al., 1989) networks. All these networks may be downloaded from the Causal Explorer webpage. The Pigs, Link and Gene networks, which were also evaluated in Tsamardinos et al. (2006a), are omitted from our experiment due to memory and



(a) (b)

Figure 14: Cumulative numbers of (a) CI tests and (b) log operations required by PC, TPDA, and RAI for learning the Alarm network. Different gray shades represent different sizes of condition sets. Percentages on tops of the bars are with reference to the RAI algorithm.

run-time limitations of the platform used in our experiment. These limitations are in the computation of the BDeu scoring function (part of the BNT toolbox) that is used for selecting a threshold for the RAI CI tests (Section 4.2).

The Casual Explorer webpage also contains larger networks that were created by tiling networks, such as the Alarm, Hailfinder, Child and Insurance, 3, 5 and 10 times. In the tiling method developed by Tsamardinos et al. (2006b), several copies (here 3, 5 and 10) of the same BN are tiled until reaching a network having a desired number of variables (e.g., Alarm5 has $5 \times 37 = 185$ variables). The method maintains the structural and probabilistic properties of the original network but allows the evaluation of the learning algorithm as the number of variables increases without increasing the complexity of the network. Overall, we downloaded and used nineteen networks, the most important details of which are shown in Table 2. Further motivation for using these networks and tiling is given in Tsamardinos et al. (2006a).

Throughout this experiment, we used for each network the same training and test sets as used in Tsamardinos et al. (2006a), so we could compare the performance of the RAI to all the algorithms reported in Tsamardinos et al. (2006a). The data in the Causal Explorer webpage are given for each network using five training sets and five test sets with 500, 1000 and 5,000 samples each. We picked and downloaded the data sets with the smallest sample size (500), which we believe challenge the algorithms the most. All the reported results for a network and a learning algorithm in this subsection are averages over five experiments in which a different training set was used for training the learning algorithm and a different test set was used for testing this algorithm.

The RAI algorithm was run by us. CMI thresholds for CI testing corresponded to the maximum BDeu values were obtained in five runs using five validation sets independent of the training and test sets, and performances were averaged over the five validation sets. We note that the thresholds selected according to the maximum BDeu values (Section 4.2) also led to the lowest SHD errors. The OR algorithm was examined with a maximum number of parents allowed for a node (k) of

#	Network	# nodes	# edges	Max fan-in	Max fan-out
1	Alarm	37	46	4	5
2	Alarm 3	111	149	4	5
3	Alarm 5	185	265	4	6
4	Alarm 10	370	570	4	7
5	Barley	48	84	4	5
6	Child	20	25	2	7
7	Child 3	60	79	3	7
8	Child 5	100	126	2	7
9	Child 10	200	257	2	7
10	Hailfinder	56	66	4	16
11	Hailfinder 3	168	283	5	18
12	Hailfinder 5	280	458	5	18
13	Hailfinder 10	560	1017	5	20
14	Insurance	27	52	3	7
15	Insurance 3	81	163	4	7
16	Insurance 5	135	281	5	8
17	Insurance 10	270	556	5	8
18	Mildew	35	46	3	3
19	Munin	189	282	3	15

Table 2: Nineteen networks with known structures that are used for the evaluation of the structure learning algorithms. The number that is attached to the network name (3, 5 or 10) indicates the number of tiles of this network. The # symbol on the first column represents the network ID for further use in the subsequent tables.

5, 10 and 20 and allowed run-time that is one and two times the time used by MMHC on the corresponding data set (OR1 and OR2, respectively). The SC algorithm was evaluated with $k = 5$ and $k = 10$ as recommended by its authors. Motivation for using these parameter values and parameter values used by the remaining algorithms are given in Tsamardinos et al. (2006a).

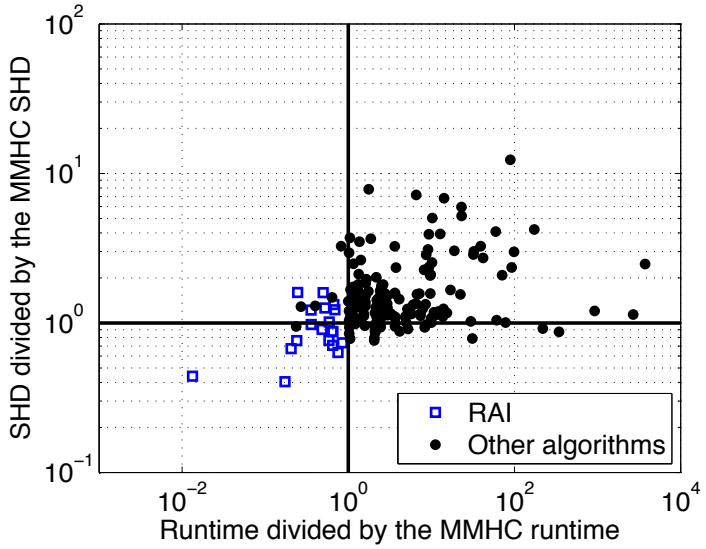
Following Tsamardinos et al. (2006a), we normalized all SHD results with the SHD results of the MMHC algorithm. For each network and algorithm, we report on the average ratio over the five runs. The normalized SHDs are presented in Table 3. A ratio smaller (larger) than 1 indicates that the algorithm learns a more (less) accurate structure than that learned using the MMHC algorithm. In addition, we average the ratios over all nineteen databases similarly to Tsamardinos et al. (2006a). Based on these averaged ratios, Tsamardinos et al. (2006a) found the MMHC algorithm to be superior to the PC, TPDA, GES, OR and SC algorithms with respect to SHD. Table 3 shows that the RAI algorithm is the only algorithm that achieves an average ratio that is smaller than 1, which means it learns structures that on average are more accurate than those learned by MMHC, and thus also more accurate than those learned by all other algorithms. Note the difference in SHD values for Alarm between Table 3 (as measured in Tsamardinos et al., 2006a, on databases of 500 samples) and Table 1 (as measured by us on databases of 10,000 samples).

#	MMHC	OR1 $k=5$	OR1 $k=10$	OR1 $k=20$	OR2 $k=5$	OR2 $k=10$	OR2 $k=20$	SC $k=5$	SC $k=10$	GS	PC	TPDA	GES	RAI
1	1.00	1.23	1.39	1.67	1.05	1.02	1.40	1.63	1.66	2.02	3.66	2.34		1.23
2	1.00	1.85	1.95	1.96	1.78	1.77	1.80	1.57	1.57	2.26	2.49	3.94		1.26
3	1.00	1.59	1.61	1.63	1.48	1.63	1.69	1.32	1.35	2.10	2.35	3.10		1.02
4	1.00	1.46	1.52	1.53	1.49	1.52	1.57	1.18		2.09		2.72		0.87
5	1.00	1.03	1.05	1.08	0.98	0.97	0.99	1.15		1.16	12.34	1.44	0.92	0.67
6	1.00	1.38	1.30	1.15	1.25	1.24	1.15	1.48	1.56	0.79	3.26	7.18	0.79	1.60
7	1.00	0.99	1.06	1.03	0.87	0.86	1.01	0.95	0.97	0.94	2.95	5.03	1.20	1.22
8	1.00	1.45	1.74	1.69	0.89	1.10	0.99	0.88	0.93	1.15	3.71	6.82	2.48	1.59
9	1.00	2.12	1.40	1.81	1.42	1.44	1.45	1.08	1.12	1.19	3.49	5.96		1.33
10	1.00	1.01	0.99	1.03	0.99	0.99	1.01	0.96		0.99	2.64	2.36	1.14	0.41
11	1.00	1.33	1.34	1.34	1.27	1.26	1.28	1.10		1.01	3.92	3.01		0.71
12	1.00	1.40	1.41	1.42	1.30	1.30	1.28	1.12		1.01	5.20	3.26		0.76
13	1.00	1.33	1.33	1.34	1.34	1.29	1.33	1.10		1.02		2.99		0.74
14	1.00	1.04	0.93	0.85	0.95	0.79	0.76	1.33	1.17	1.20	3.26	2.54	1.01	0.76
15	1.00	1.08	1.06	1.25	1.04	1.14	1.15	1.26	1.33	1.57	4.09	3.04		0.98
16	1.00	1.25	1.24	1.12	1.13	1.15	1.17	1.24	1.25	1.59	4.22	2.86		0.91
17	1.00	1.30	1.29	1.31	1.19	1.13	1.24	1.18	1.24	1.55		2.87		0.88
18	1.00	1.09	1.11	1.10	1.10	1.12	1.07	1.04		0.91	7.83	2.08	0.87	0.63
19	1.00	1.09	1.16	1.06	1.17			0.95		1.30		1.29		0.44
avg.	1.00	1.32	1.31	1.33	1.19	1.21	1.24	1.19	1.29	1.36	4.36	3.41	1.20	0.95

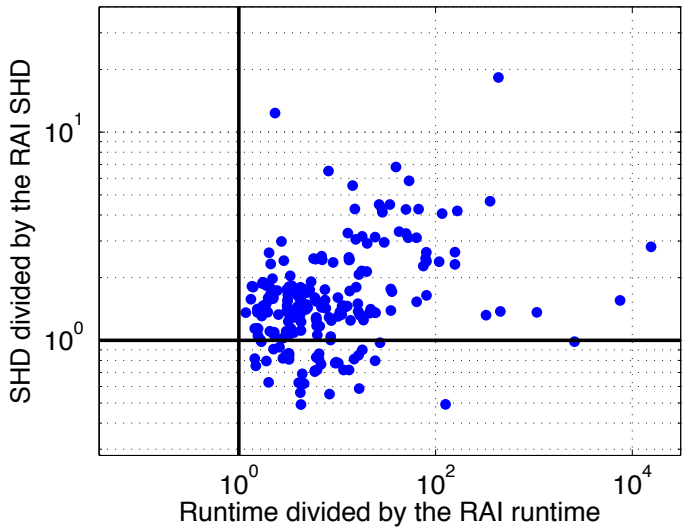
Table 3: Algorithm SHD errors normalized with respect to the MMHC SHD error for the nineteen networks detailed in Table 2. Average (avg.) for an algorithm is over all networks. Blank cells represent jobs that Tsamardinos et al. (2006a) reported that refused to run or did not complete their computations within two days running time.

Next, we compared the run-times of the algorithms in learning the nineteen networks. We note that the run-time of a structure learning algorithm depends, besides on its implementation, on the number of statistical calls (Tsamardinos et al., 2006a) it performs (e.g., CI tests in CB algorithms). For CB algorithms it also depends on the orders of the CI tests and the number of states of each variable that is included in the condition set. The run-time for each algorithm learning each network is presented in Table 4. Following Tsamardinos et al. (2006a), we normalized all run-time results with the run-time results of the MMHC algorithm and report on the average ratio for each algorithm and network over five runs. The run-time ratios for all algorithms except that for the RAI were taken from the Causal Explorer webpage. The ratio for the RAI was computed after running both the RAI and MMHC algorithms on our platform using the same data sets. According to Tsamardinos et al. (2006a), MMHC is the fastest algorithm among all algorithms (except RAI). Table 4 shows that RAI was the only algorithm that achieved an average ratio smaller than 1, which means it is the new fastest algorithm. The RAI average run-time was between 2.1 (for MMHC) and 2387 (for GES) times shorter than those of all other algorithms. Perhaps part of the inferiority of GES with respect to run-time can be related (Tsamardinos et al., 2006a) to many optimizations suggested in Chickering (2002) that were not implemented in Tetrad 4.3.1 that was used by Tsamardinos et al. (2006a) affecting their, and thus also our, results.

Accounting for both error and time, we plot in Figure 15 the SHD and run-time for all nineteen networks normalized with respect to either the MMHC algorithm (Figure 15a) or the RAI algorithm



(a)



(b)

Figure 15: Normalized SHD vs. normalized run-time for all algorithms learning all networks. (a) Normalization is with respect to the MMHC algorithm (thus MMHC results are at (1,1)) and (b) normalization is with respect to the RAI algorithm (thus RAI results are at (1,1)). The points in the graph correspond to 19 networks (average performance over 5 runs) and $14-1=13$ algorithms.

#	MMHC	OR1 $k=5$	OR1 $k=10$	OR1 $k=20$	OR2 $k=5$	OR2 $k=10$	OR2 $k=20$	SC $k=5$	SC $k=10$	GS	PC	TPDA	GES	RAI
1	1.00	1.14	1.00	1.07	2.24	2.22	2.33	1.75	16.93	2.17	1.87	3.74		0.69
2	1.00	1.62	1.65	1.64	2.51	2.53	2.63	7.15	9.71	8.16	1.15	12.75		0.52
3	1.00	1.21	1.32	1.33	2.35	2.41	2.48	6.01	6.54	9.80	92.64	9.11		0.59
4	1.00	1.38	1.61	1.43	2.87	2.93	2.77	13.85		71.15		41.81		0.65
5	1.00	1.26	1.24	1.21	2.29	2.42	2.36	7.36		2.74	89.28	4.10	219.5	0.20
6	1.00	1.61	1.61	1.53	2.39	2.34	3.25	0.64	6.71	1.05	0.82	6.56	31.12	0.25
7	1.00	1.15	1.14	1.06	2.12	2.10	2.18	3.66	8.64	2.44	1.02	10.27	921	0.36
8	1.00	1.12	1.14	1.13	2.10	2.19	2.29	4.16	8.31	5.76	1.05	14.19	3738	0.50
9	1.00	1.34	1.05	1.32	2.20	2.28	2.45	9.97	11.08	12.10	1.36	22.99		0.67
10	1.00	1.20	1.22	1.21	2.31	2.29	2.28	1.58		1.04	1.42	9.31	2690	0.17
11	1.00	1.13	1.15	1.14	2.15	2.21	2.27	4.88		4.96	9.32	32.39		0.65
12	1.00	1.11	1.15	1.17	2.24	2.27	2.19	7.39		10.01	23.14	39.22		0.58
13	1.00	1.18	1.19	1.15	2.94	2.61	2.74	13.77		29.84		99.00		0.85
14	1.00	1.02	1.03	1.03	2.09	2.06	2.05	1.26	15.36	1.02	3.62	10.19	78.06	0.24
15	1.00	1.09	1.13	1.18	2.25	2.38	2.21	2.96	8.50	3.63	59.50	18.87		0.36
16	1.00	1.49	1.48	1.54	2.97	2.95	2.96	5.15	7.88	3.63	173.3	8.67		0.48
17	1.00	1.19	1.12	1.20	2.30	2.35	2.40	10.73	13.95	22.34		32.00		0.64
18	1.00	2.46	2.43	2.55	3.68	3.46	3.68	61.04		5.23	1.76	9.67	343.7	0.75
19	1.00	1.05	1.07	1.08	2.09			0.24		0.40		0.27		0.01
avg.	1.00	1.30	1.30	1.31	2.43	2.45	2.53	8.61	10.33	10.39	30.75	20.27	1146	0.48

Table 4: Algorithm run-times normalized with respect to the MMHC run-time for the nineteen networks detailed in Table 2. Average (avg.) for an algorithm is over all networks. Blank cells represent jobs that Tsamardinos et al. (2006a) reported that refused to run or did not complete their computations within two days running time.

(Figure 15b). Figure 15 demonstrates that the advantage of RAI over all other algorithms is evident for both the SHD error and the run-time.

It is common to consider the statistical calls performed by an algorithm of structure learning as the major criterion of computational complexity (efficiency) and a major contributor to the algorithm run-rime. In CB algorithms (e.g., PC, TPDA and RAI), the statistical calls are due to CI tests, and in S&S algorithms (e.g., GS, GES, SC, OR) the calls are due to the computation of the score. Hybrid algorithms (e.g., MMHC) have both types of calls. In Table 5, we compare the numbers of calls for statistical tests performed by the RAI algorithm and computed by us to those of the MMHC, GS, PC and TPDA, as computed in Tsamardinos et al. (2006a), and downloaded from the Causal Explorer webpage. We find that for all networks the RAI algorithm performs fewer calls for statistical tests than all other algorithms. On average over all networks, the RAI algorithm performs only 53% of the calls for statistical tests performed by the MMHC algorithm, which is the algorithm that required the fewest calls of all algorithms examined in Tsamardinos et al. (2006a). Figure 16 demonstrates this advantage of RAI over MMHC graphically using a scatter plot. All points below the $x = y$ line represent data sets for which the numbers of calls for statistical tests of MMHC are larger than those of RAI.

Evaluating the statistical significance of the results in Tables 3-5 using Wilcoxon signed-ranks test (Demšar, 2006) with a confidence level of 0.05, we find the SHD errors of RAI and MMHC to be not significantly different from each other; however, the RAI run-times and numbers of statistical calls are significantly shorter than those of the MMHC algorithm.

#	MMHC	GS	PC	TPDA	RAI
1	1.00	2.42	9.95	1.94	0.81
2	1.00	3.78	2.51	3.34	0.57
3	1.00	4.44	1499.22	3.02	0.67
4	1.00	5.12		2.64	0.75
5	1.00	1.96	2995.87	1.58	0.34
6	1.00	1.32	3.61	2.92	0.21
7	1.00	2.49	4.61	2.97	0.39
8	1.00	3.25	4.40	3.17	0.51
9	1.00	3.91	5.43	3.13	0.64
10	1.00	1.75	36.54	1.93	0.30
11	1.00	2.57	340.44	1.83	0.72
12	1.00	3.07	1033.86	1.87	0.67
13	1.00	3.40		1.85	0.77
14	1.00	1.32	40.57	2.97	0.27
15	1.00	2.35	1082.45	2.71	0.39
16	1.00	3.12	5143.51	2.97	0.49
17	1.00	4.25		3.20	0.63
18	1.00	3.38	10.78	3.49	0.59
19	1.00	1.75		0.91	0.30
avg.	1.00	2.93	814.25	2.55	0.53

Table 5: Number of statistical calls performed by each algorithm normalized by the number of statistical calls performed by the MMHC algorithm for the nineteen networks detailed in Table 2. Average (avg.) for an algorithm is over all networks. Blank cells represent jobs that Tsamardinos et al. (2006a) reported that refused to run or did not complete their computations within two days running time.

In continuation to Section 4.1, we further analyzed the complexity of RAI (as measured by the numbers of CI tests performed) according to the CI test orders and the graph size. However, here we used real rather than synthetic data. We examined the numbers of tests as performed for different orders for the Child, Insurance, Alarm and Hailfinder networks and their tiled networks. Using the tiled networks (Tsamardinos et al., 2006b), we could examine the impact of graph size on the number of tests. Figure 17 shows the cumulative percentage of CI tests for a specific order out of the total number of CI tests performed for each network. The figure demonstrates that the percentages of CI tests performed decrease with the CI test order and become small for orders higher than the max fan-in of the network (see Table 2). These percentages also decrease with the numbers of nodes in the network (validated on the tiled networks). This is due to a faster increase of the number of low-order CI tests compared with the number of high-order CI tests as the graph size increases for all networks except for Hailfinder. For Hailefinder (Figure 17d), the threshold for the network was different from those of the tiled networks. This led to an increase in the percentage of high-order CI tests and a decrease in CI tests of order 0 when comparing the Hailfinder network to its tiled versions. For all the tiled Alarm networks (Figure 17c), CI tests of order 0 nearly sufficed

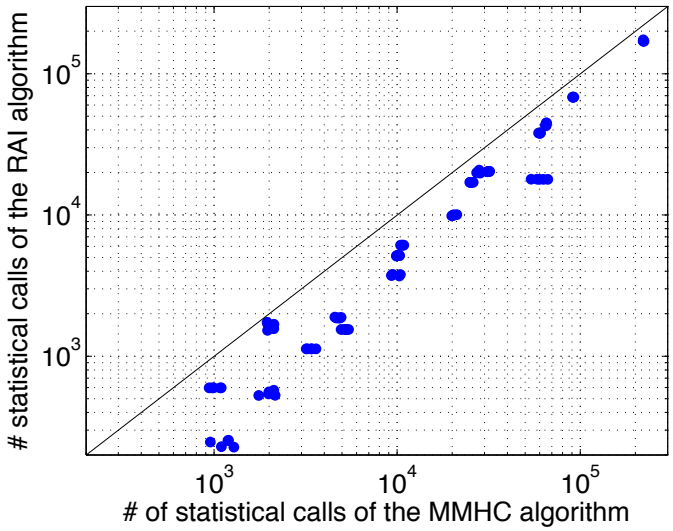


Figure 16: Number of statistical calls performed by the RAI algorithm vs. the number of statistical calls performed by the MMHC algorithm for all networks and data sets examined in this sub-section (5 data sets $\times$ 19 networks = 95 points).

for learning the network. Overall, the results support our preliminary results with synthetic data and "perfect" CI tests (Section 4.1). Thus, we can conclude that as the graph size increases, the RAI algorithm requires relatively fewer CI tests of high orders, especially of orders higher than the max fan-in, than tests of low orders. This result enhances the attractiveness in applying the RAI algorithm also to large problems.

4.5 Structure Learning for General BN Classifiers

Classification is one of the most fundamental tasks in machine learning (ML), and a classifier is primarily expected to achieve high classification accuracy. The Bayesian network classifier (BNC) is usually not considered as an accurate classifier compared to state-of-the-art ML classifiers, such as the neural network (NN) and support vector machine (SVM). However, the BNC has important advantages over the NN and SVM models. The BNC enhances model interpretability by exhibiting dependences, independences and causal relations between variables. It also allows the incorporation of prior knowledge during model learning so as to select a better model or to improve the estimation of its data-driven parameters. Moreover, the BNC naturally performs feature selection as part of model construction and permits the inclusion of hidden nodes that increase model representability and predictability. In addition, the BN has a natural way of dealing with missing inputs by marginalizing hidden variables. Finally, compared to NN and SVM, BNC can model very large, multi-class problems with different types of variables. These advantages are important in real-world classification problems, since they provide many insights into the problem at hand that are beyond the pure classification decisions provided by NN and SVM.

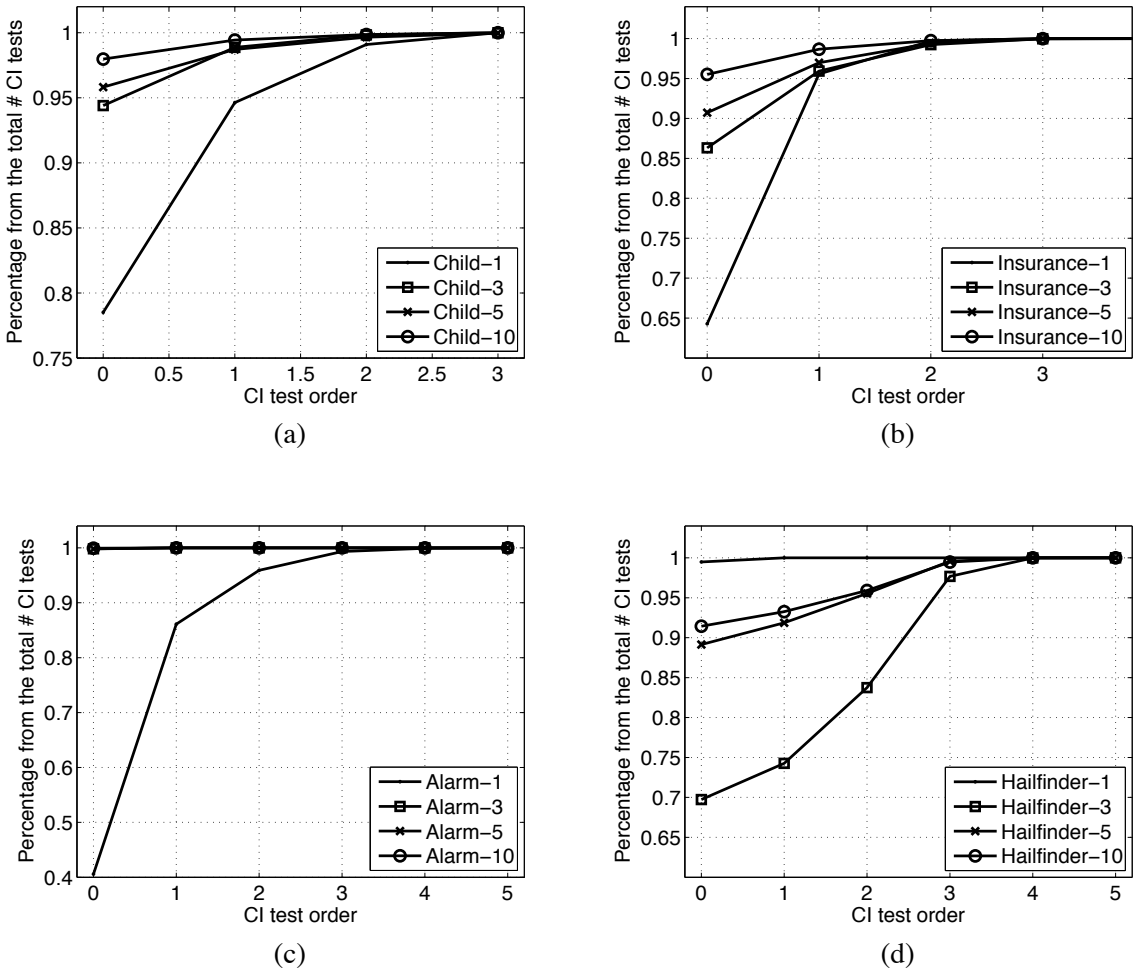


Figure 17: Cumulative percentages of CI tests out of the total numbers of tests for increasing orders as performed by the RAI algorithm for the (a) Child, (b) Insurance, (c) Alarm, and (d) Hailfinder networks including their tiled networks.

We evaluated the RAI complexity, run-time and accuracy when applied to learning a general BN classifier (Cheng and Greiner, 1999; Friedman et al., 1997) in comparison to other algorithms of structure learning using nineteen databases of the UCI Repository (Newman et al., 1998) and Kohavi and John (1997). These databases are detailed in Table 6 with respect to the numbers of variables, classes and instances in each database. All databases were analyzed using a CV5 experiment, except large databases (e.g., "chess", "nursery" and "shuttle"), which were analyzed using the holdout methodology and the common division to training and test sets (Newman et al., 1998; Friedman et al., 1997; Cheng et al., 1997) as detailed in Table 6. Continuous variables were discretized using the MLC++ library (Kohavi et al., 1994) and instances with missing values were removed, as is commonly done.

Database	# variables	# classes	# instances	Test methodology	# training instances	# test instances
australian	14	2	690	CV5	552	138
breast	9	2	683	CV5	544	136
car	6	4	1728	CV5	1380	345
chess	36	2	3196	holdout	2130	1066
cleve	11	2	296	CV5	236	59
cmc	9	3	1473	CV5	1176	294
corral	6	2	128	CV5	100	25
crx	15	2	653	CV5	520	130
flare C	10	9	1389	CV5	1108	277
iris	4	3	150	CV5	120	30
led7	7	10	3200	CV5	2560	640
mofn 3-7-10	10	2	1324	holdout	300	1024
nursery	8	5	12960	holdout	8640	4320
shuttle (s)	8	7	5800	holdout	3866	1934
tic-tac-toe	9	2	958	CV5	764	191
vehicle	18	4	846	CV5	676	169
vote	16	3	435	CV5	348	87
wine	13	3	178	CV5	140	35
zoo	16	7	101	CV5	80	20

Table 6: Databases of the UCI repository (Newman et al., 1998) and of Kohavi and John (1997) used for evaluating the accuracy of a classifier learned using the RAI algorithm.

Generally for this sub-section, CI tests for RAI and PC were carried out using the χ^2 test (Spirtes et al., 2000) and those for TPDA using the CMI independence test (Equation 2). However, CI tests for RAI and PC for the "corral", "nursery" and "vehicle" databases were carried out using the CMI independence test. In the case of the large "nursery" database, the need to use the CMI test was due to a Matlab memory limitation in the completion of the χ^2 test using the BNT structure learning package (Leray and François, 2004). In the case of the "corral" and "vehicle" databases, the smallness of the database, together with either the large numbers of classes, variables or states for each variable, led to low frequencies of instances for many combinations of variable states. In this case, the implementation of the χ^2 test assumes variable dependence (Spirtes et al., 2000) that prevents the CB (PC, TPDA and RAI) algorithms from removing edges regardless of the order of the CI test, leading to erroneous decisions. Another test of independence, which is reported to be more reliable and robust, especially for small databases or large numbers of variables (Dash and Druzdzel, 2003), may constitute another solution in these cases.

Thresholds for the CI tests of the CB algorithms and parameter values for all other algorithms were chosen for each algorithm and database so as to maximize the classification accuracy on a validation set selected from the training set or based on the recommendation of the algorithm authors or of Tsamardinos et al. (2006a). Although using a validation set decreases the size of the training set, it also eliminates the chance of selecting a threshold or a parameter that causes the model to

overfit the training set at the expense of the test set. If several thresholds/parameters were found suitable for an algorithm, the threshold/parameter chosen was that leading to the fewest CI tests (in the case of CB algorithms). For GES and GS there are no parameters to set (except the equivalent sample size for the BDeu), and for MMHC we used the selections used by the authors in all their experiments.

Finally, parameter learning was performed by maximum likelihood estimation. Since we were interested in structure learning, no attempt was made to study estimation methods other than this simple and most popular generative method (Cooper and Herskovits, 1992; Heckerman, 1995; Yang and Chang, 2002). Nevertheless, we note that discriminative models for parameter learning have recently been suggested (Pernkopf and Bilmes, 2005; Roos et al., 2005). These models show an improvement over generative models when estimating the classification accuracy (Pernkopf and Bilmes, 2005). We expect that any improvement in classification accuracy gained by using parameter learning other than maximum likelihood estimation will be shared by classifiers induced using any algorithm of structure learning; however, the exact degree of improvement in each case should be further evaluated.

Complexity of the RAI algorithm was measured by the number of CI tests employed for each size of the condition set and the cumulative run-time of the CI tests. These two criteria of complexity were also measured for the PC algorithm, since both the RAI and PC algorithms use the same implementation of CI testing. Table 7 shows the average number and percentage of CI tests reduced by the RAI algorithm compared to the PC algorithm for different CI test orders and each database. An empty entry in the table means that no CI tests of this order are required. A 100% cut in CI tests for a specific order means that RAI does not need any of the CI tests employed by the PC algorithm for this order (e.g., orders 2 and above for the "led7" database). It can be seen that for almost all databases examined, the RAI algorithm avoids most of the CI tests of orders two and above that are required by the PC algorithm (e.g., the "chess" database). Table 7 also shows the reduction in the CI test run-time due to the RAI algorithm in comparison to the PC algorithm for all nineteen databases examined; except for the "australian" database, the cut is measured in tens of percentages for all databases and for six databases this cut is higher than 70%. Run-time differences between algorithms may be the result of different implementations. However, since in our case the run-time is almost entirely based on the number and order of CI tests and RAI has reduced most of the PC CI tests, especially those of high orders that are expensive in run-time, we consider the above run-time reduction results to be significant.

Classification accuracy using a BNC has recently been explored extensively in the literature (Friedman et al., 1997; Grossman and Domingos, 2004; Kontkanen et al., 1999; Pernkopf and Bilmes, 2005; Roos et al., 2005). By restricting the general inference task of BN to inference performed on the class variable, we turn a BN into a BNC. First, we use the training data to learn the structure and then transform the pattern outputted by the algorithm into a DAG (Dor and Tarsi, 1992). Thereafter, we identify the class node Markov blanket and remove from the graph all the nodes that are not part of this blanket. Now, we could estimate the probabilities comprising the class node posterior probability, $P(C|\boldsymbol{X})$, where $\boldsymbol{X}$ is the set of the Markov blanket variables. During the test, we inferred the state c of the class node C for each test instantiation, $\boldsymbol{X} = \boldsymbol{x}$, using the estimated posterior probability. The class $\hat{c}$ selected was the one that maximized the posterior probability, meaning that $\hat{c} = \arg\max_c P(C = c|\boldsymbol{X} = \boldsymbol{x})$. By comparing the class maximizing the posterior probability and the true class, we could compute the classification accuracy.

Database	CI test order								Run-time cut (%)
	0	1		2		3		4	
australian	0 (0)	3.8	(34.4)						6.05
breast	0 (0)	107.2	(54.8)	35	(99.1)				71.87
car	0 (0)	16	(100)	11.2	(100)	3.2	(100)		91.10
chess	0 (0)	2263	(76.3)	2516	(89)	581	(94)	249	(100) 80.65
cleve	0 (0)	12.4	(63)						39.60
cmc	0 (0)	10.2	(10.9)	8	(32.5)				14.22
corral	0 (0)	22.4	(100)	26	(100)	3.6	(100)		87.94
crx	0 (0)	8.8	(49.6)						25.25
flare C	0 (0)	16	(39.6)	3	(100)				20.38
iris	0 (0)	2	(40)						19.10
led7	0 (0)	46.2	(45.7)	105	(100)	140	(100)	105	(100) 91.74
mofn 3-7-10	0 (0)	17	(100)	4	(100)				67.70
nursery	0 (0)	20	(100)	30	(100)	20	(100)	5	(100) 89.70
shuttle (s)	0 (0)	1.4	(0.7)	95.8	(43.8)	117.6	(49.3)	83.6	(56.0) 38.94
tic-tac-toe	0 (0)	53.2	(27.1)	56.6	(48.6)	1.8	(51.4)		36.52
vehicle	0 (0)	-12.4	(-2.9)	32.6	(20.4)	-5.8	(-14.0)	3.4	(27.4) 13.15
vote	0 (0)	24.2	(21.9)	17.2	(98.1)	6.4	(100)	1	(100) 46.06
wine	0 (0)	25.8	(41.0)	44.2	(67.6)	40.6	(82.4)	19	(96.7) 29.11
zoo	0 (0)	82	(27.8)	365.8	(29.6)	1033.4	(27.7)	1928.6	(25.6) 13.63

Table 7: Average number (and percentage) of CI tests reduced by the RAI algorithm compared to the PC algorithm for different databases and CI test orders and the cut (%) in the total CI test run-time.

In Table 8 we compared the classification accuracy due to the RAI algorithm to those due to the PC, TPDA, GES, MMHC, SC and NBC algorithms. We note the overall advantage of the RAI algorithm, especially for large databases. Since the reliability of the CI tests increased with the sample size, it seems that RAI benefits from this increase more than the other algorithms and excels in classifying large databases. RAI, when compared to the other structure learning algorithms, yielded the best classifiers on six ("flare C", "nursery", "led7", "mofn", "tic-tac-toe" and "vehicle") of the ten largest databases and among the best classifiers on the remaining four ("shuttle", "chess", "car" and "cmc") large databases. The other CB algorithms—PC and TPDA—also showed here, and in Tsamardinos et al. (2006a), better results on the large databases. However, the CB algorithms are less accurate on very small databases (e.g., "wine" and "zoo").

Overall, RAI was the best algorithm on 7 databases compared to 5, 2, 5, 4, 5 and 5 databases for the PC, TPDA, GES, MMHC, SC and NBC algorithms, respectively. RAI was the worst classifier on only a single database, whereas the PC, TPDA, GES, MMHC, SC and NBC algorithms were the worst classifiers on 2, 4, 6, 2, 2 and 7 databases, respectively. We believe that the poor results of the GES and MMHC algorithms on the "nursery" database may be attributed to the fact that these algorithms find the class node C as a child of many other variables, making the estimation of $P(C|\boldsymbol{X})$ unreliable due to the curse-of-dimensionality. The structures learned by the other algorithms required a smaller number of such connections and thereby reduced the curse.

Database	PC	TPDA	GES	MMHC	SC	NBC	RAI
australian	85.5 (0.5)	85.5 (0.5)	*83.5* (2.1)	**86.2** (1.5)	85.5 (1.2)	85.9 (3.4)	85.5 (0.5)
breast	95.5 (2.0)	*94.4* (2.7)	96.8 (1.1)	97.2 (1.2)	96.5 (0.8)	**97.5** (0.8)	96.5 (1.6)
car	84.3 (2.6)	84.5 (0.6)	*81.5* (2.3)	90.2 (2.0)	**93.8** (1.1)	84.7 (1.3)	92.9 (1.1)
chess	93.1	90.1	**97.0**	94.1	92.5	*87.1*	93.5
cleve	76.7 (7.2)	*72.0* (10.7)	79.4 (5.7)	82.1 (4.5)	**83.5** (5.7)	**83.5** (5.2)	81.4 (5.4)
cmc	50.9 (2.3)	46.4 (2.1)	*46.3* (1.5)	48.6 (2.6)	49.7 (2.5)	**51.3** (1.3)	51.1 (3.2)
corral	**100** (0)	88.2 (6.4)	**100** (0)	**100** (0)	**100** (0)	*85.2* (7.3)	**100** (0)
crx	86.4 (2.6)	**86.7** (3.4)	*82.2* (6.4)	**86.7** (1.7)	**86.7** (3.4)	86.2 (2.8)	86.4 (2.6)
flare C	**84.3** (2.5)	**84.3** (2.4)	**84.3** (2.5)	**84.3** (2.5)	**84.3** (2.5)	*77.7* (3.1)	**84.3** (2.5)
iris	**96.0** (4.3)	93.3 (2.4)	**96.0** (4.3)	94.0 (3.6)	*92.7* (1.5)	94.0 (4.3)	93.3 (2.4)
led7	73.3 (1.8)	*72.9* (1.5)	*72.9* (1.5)	*72.9* (1.5)	*72.9* (1.5)	*72.9* (1.5)	**73.6** (1.6)
mofn 3-7-10	81.4	90.8	*79.8*	90.5	91.9	89.8	**93.2**
nursery	**72.0**	64.7	33.3	*29.3*	30.3	66.0	**72.0**
shuttle (s)	98.4	*96.3*	**99.5**	99.2	99.2	98.8	99.2
tic-tac-toe	74.7 (1.4)	72.2 (3.8)	69.9 (2.8)	71.1 (4.2)	70.4 (4.7)	*69.6* (3.1)	**75.6** (1.9)
vehicle	63.9 (3.3)	65.6 (2.8)	64.1 (11.2)	69.3 (1.5)	64.8 (9.1)	*62.0* (4.0)	**70.2** (2.8)
vote	**95.9** (1.5)	95.4 (2.1)	94.7 (2.8)	95.6 (2.2)	93.1 (2.2)	*90.6* (3.3)	95.4 (1.6)
wine	*85.4* (7.8)	97.8 (3.0)	98.3 (2.5)	98.3 (2.5)	98.3 (2.5)	**98.9** (1.5)	87.1 (5.9)
zoo	*89.0* (8.8)	96.1 (2.2)	96.0 (2.3)	93.1 (4.5)	95.9 (6.9)	**96.3** (3.8)	*89.0* (8.79)
average	83.5	83.0	*81.9*	83.3	83.3	83.1	**85.3**
std	12.7	13.8	18.4	18.4	18.4	13.3	12.3

Table 8: Mean (and standard deviation for CV5 experiments) of the classification accuracy of the RAI algorithm in comparison to those of the PC, TPDA, GES, MMHC, SC and NBC algorithms. **Bold** and *italic* fonts represent, respectively, the best and worst classifiers for a database.

In addition, we averaged the classification accuracies of the algorithms over the nineteen databases. Averaging accuracies over databases has no meaning in itself except that the average accuracies over many different problems of different algorithms may infer about the relative expected success of the algorithms in other classification problems. It is interesting to note that although the different algorithms in our study showed different degrees of success on various databases, most of the algorithms (i.e., PC, TPDA, MMHC, SC and NBC) achieved almost the same average accuracy (83.0%-83.5%). The GES average accuracy was a little inferior (81.9%) to that of the above algorithms, and the average accuracy of the RAI (85.3%) was superior to that of all algorithms. Concerning the standard deviation of the classification accuracy, RAI outperformed all classifiers implying to the robustness of the RAI-based classifier.

Superiority of one algorithm over another algorithm for each database was evaluated with a statistical significance test (Dietterich, 1998). We used a single-sided t-test to evaluate whether the mean difference between any pair of algorithms as measured on the five folds of the CV5 test was greater than zero. Table 9 summarizes the statistical significance results, measured at a significance level of 0.05, for any two classifiers and each database examined using cross validation. The number in each cell of Table 9 describes—for the corresponding algorithm and database—the number of

Databse	PC	TPDA	GES	MMHC	SC	NBC	RAI
australian	1	1	0	1	0	1	1
breast	0	0	0	2	0	3	0
car	1	1	0	4	6	1	5
cleve	0	0	0	1	3	3	2
cmc	4	0	0	1	2	2	3
corral	2	0	2	2	2	0	2
crx	0	0	0	0	0	0	0
flare C	1	1	1	1	1	0	1
iris	1	0	1	0	0	0	0
led7	0	0	0	0	0	0	5
tic-tac-toe	3	2	0	0	0	0	5
vehicle	0	1	0	3	0	0	3
vote	2	2	1	3	0	0	1
wine	0	2	2	2	2	2	0
zoo	0	0	0	0	2	0	0
total	15	10	7	20	18	12	**28**
average	1.00	0.67	0.47	1.33	1.20	0.8	**1.87**

Table 9: Statistical significance using a t-test for the classification accuracy results of Table 8. For a given database, each cell indicates the number of algorithms found to be inferior at a significance level of 0.05 to the algorithm above the cell.

algorithms that are inferior to that algorithm for that databases. A "0" value indicates that the algorithm is either inferior to all the other algorithms or not significantly superior to any of them. For example, for the "car" database the PC, TPDA, GES, MMHC, SC, NBC and RAI algorithms were significantly superior to 1, 1, 0, 4, 6, 1 and 5 other algorithms, respectively. In total, the superiority of the RAI algorithm over the other algorithms was statistically significant 28 times, with an average of 1.87 algorithms per database. The second and third best algorithms were the MMHC and SC algorithms, with a total of 20 and 18 times of statistically significant superiority and averages of 1.33 and 1.2 per database, respectively. The least successful classifier, according to Tables 8 and 9, was the one that is learned using GES. We believe that this inferiority arises from the assumptions on the type of probabilities and their parameters made by the GES algorithm when computing the BDeu score (Heckerman et al., 1995), assumptions that probably do not hold for the examined databases.

Although this methodology of statistical tests between pairs of classifiers is the most popular in the machine learning community, there are other methodologies that evaluate statistical significance between several classifiers on several databases simultaneously. For example, Demšar (2006), recently suggested using Friedman test (Friedman, 1940) and some post-hoc tests for such an evaluation.

5. Discussion

The performance of a CB algorithm in BN structure learning depends on the number of conditional independence tests and the sizes of condition sets involved in these tests. The larger the condition set, the greater the number of CI tests of high orders that have to be performed and the smaller their accuracies.

We propose the CB RAI algorithm that learns a BN structure by performing the following sequence of operations: 1) test of CI between nodes and removal of edges related to independences, 2) edge direction employing orientation rules, and 3) structure decomposition into smaller autonomous sub-structures. This sequence of operations is performed recursively for each sub-structure, along with increasing the order of the CI tests. Thereby, the RAI algorithm deals with less potential parents for the nodes on a tested edge and thus uses smaller condition sets that enable the performance of fewer CI tests of higher orders. This reduces the algorithm run-time and increases its accuracy.

By introducing orientation rules through edge direction in early stages of the algorithm and following CI tests of lower orders, the graph "backbone" is established using the most reliable CI tests. Relying on this "backbone" and its directed edges in later stages obviates the need for unnecessary CI tests and enables RAI to be less complex and sensitive to errors.

In this study, we proved the correctness of the RAI algorithm. In addition, we demonstrated empirically, using synthetically generated networks, samples of nineteen known structures, and nineteen natural databases used in classification problems, the advantage of the RAI algorithm over state-of-the-art structure learning algorithms, such as PC, TPDA, GS, GES, OR, SC and MMHC, with respect to structural correctness, number of statistical calls, run-time and classification accuracy. We note that no attempt was made to optimize the parameters of the other algorithms and the effect of such optimization was not evaluated. This is due to the fact that some of the algorithms have more than one parameter to optimize and besides, no optimization methods were proposed by the algorithm inventors. We propose such an optimization method for the RAI algorithm that uses only the training (validation) data.

We plan to extend our study in several directions. One is the comparison of RAI-based classifiers to non-BN classifiers, such as the neural network and support vector machine. Second is the incorporation of different types of prior knowledge (e.g., related to classification) into structure learning. We also intend to study error correction during learning and to allow the inclusion of hidden variables to improve representation and facilitate learning with the RAI algorithm.

Acknowledgments

The authors thank the three anonymous reviewers for their thorough reviews and helpful comments that improved the quality and clarity of the manuscript. The authors also thank the Discovery Systems Laboratory (DSL) of the Vanderbilt University, TN, for making the Causal Explorer library of algorithms and the networks tested in Aliferis et al. (2003) freely available. Special thanks due to Ms. Laura Brown of DSL for the co-operation, helpful discussions and the provision of some missing results of Aliferis et al. (2003) for comparison. This work was supported, in part, by the Paul Ivanier Center for Robotics and Production Management, Ben-Gurion University of the Negev, Beer-Sheva, Israel.

References

C. F. Aliferis, I. Tsamardinos, A. Statnikov, and L. E. Brown. Causal Explorer: A causal probabilistic network learning toolkit for biomedical discovery. In *Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, pages 371–376, 2003.

S. Andreassen, F. V. Jensen, S. K. Andersen, B. Falck, U. Kjærulff, M. Woldbye, A. R. Sørensen, A. Rosenfalck, and F. Jensen. MUNIN—an expert EMG assistant. In John E. Desmedt, editor, *Computer-Aided Electromyography and Expert Systems*, chapter 21, pages 255–277. Elsevier Science Publishers, 1989.

I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, pages 246–256, 1989.

J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.

J. Cheng. PowerConstructor system. `http://www.cs.ualberta.ca/~jcheng/bnpc.htm`, 1998.

J. Cheng and R. Greiner. Comparing Bayesian network classifiers. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 101–107, 1999.

J. Cheng, D. Bell, and W. Liu. Learning Bayesian networks from data: An efficient approach based on information theory. In *Proceedings of the Sixth ACM International Conference on Information and Knowledge Management*, pages 325–331, 1997.

J. Cheng, C. Hatzis, H. Hayashi, M. Krogel, S. Morishita, D. Page, and J. Sese. KDD cup 2001 report. *ACM SIGKDD Explorations Newsletter*, 3:47–64, 2002.

D. M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

G. F. Cooper and E. A. Herskovits. Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

R. G. Cowell. Conditions under which conditional independence and scoring methods lead to identical selection of Bayesian network models. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 91–97, 2001.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

D. Dash and M. Druzdzel. A hybrid anytime algorithm for the construction of causal models from sparse sata. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 142–149, 1999.

D. Dash and M. Druzdzel. Robust independence testing for constraint-based learning of causal structure. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 167–174, 2003.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

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

D. Dor and M. Tarsi. A simple algorithm to construct a consistent extension of a partially oriented graph. Technical Report R-185, Cognitive Systems Laboratory, UCLA Computer Science Department, 1992.

M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11:86–92, 1940.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29: 131–161, 1997.

N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive datasets: The "sparse-candidate" algorithm. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 206–215, 1999.

D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 361–368, 2004.

D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report TR-95-06, Microsoft Research, 1995.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

D. Heckerman, C. Meek, and G. F. Cooper. A Bayesian approach to causal discovery. In G. Glymour and G. Cooper, editors, *Computation, Causation and Discovery*, pages 141–165. AAAI Press, 1999.

A. Jensen and F. Jensen. MIDAS—an influence diagram for management of mildew in winter wheat. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 349–356, 1996.

R. J. Kennett, K. Korb, and A. E. Nicholson. Seebreeze prediction using Bayesian networks. In *Proceedings of the Fifth Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 148–153, 2001.

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

R. Kohavi, G. H. John, R. Long, D. Manley, and K. Pfleger. MLC++: A machine learning library in C++. In *Proceedings of the Sixth International Conference on Tools with AI*, pages 740–743, 1994.

P. Kontkanen, P. Myllymaki, T. Sliander, and H. Tirri. On supervised selection of Bayesian networks. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 334–342, 1999.

K. Kristensen and I. A. Rasmussen. The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides. *Computers and Electronics in Agriculture*, 33:197–217, 2002.

S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

P. Leray and O. François. BNT structure learning package: Documentation and experiments. Technical Report FRE CNRS 2645, Laboratoire PSI, Universitè et INSA de Rouen, 2004.

M. Marengoni, C. Jaynes, A. Hanson, and E. Riseman. Ascender II, a visual framework for 3D reconstruction. In *Proceedings of the First International Conference on Computer Vision Systems*, pages 469–488, 1999.

C. Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence*, pages 403–410, 1995.

C. Meek. *Graphical Models: Selecting Causal and Statistical Models*. PhD thesis, Carnegie Mellon University, 1997.

A. Moore and W. Wong. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In *Twentieth International Conference on Machine Learning*, pages 552–559, 2003.

K. Murphy. The Bayes net toolbox for Matlab. *Computing Science and Statistics*, 33:331–350, 2001.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan–Kaufmann, 1988.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

F. Pernkopf and J. Bilmes. Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 657–664, 2005.

T. Roos, H. Wettig, P. Grunwald, P. Myllymaki, and H. Tirri. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, 59:267–296, 2005.

M. Singh and M. Valtorta. Construction of Bayesian network structures from data: A brief survey and an efficient algorithm. *International Journal of Approximate Reasoning*, 12:111–131, 1995.

P. Spirtes. An anytime algorithm for casual inference. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, pages 213–221, 2001.

P. Spirtes and C. Meek. Learning Bayesian networks with discrete variables from data. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 294–299, 1995.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. MIT Press, 2nd edition, 2000.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78, 2006a.

I. Tsamardinos, A. Statnikov, L. E. Brown, and C. F. Aliferis. Generating realistic large Bayesian networks by tiling. In *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference*, 2006b.

T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227, 1990.

S. Yang and K. C. Chang. Comparison of score metrics for Bayesian network learning. *IEEE Transactions on Systems, Man and Cybernetics A*, 32:419–428, 2002.

R. Yehezkel and B. Lerner. Recursive autonomy identification for Bayesian network structure learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 429–436, 2005.

Journal of Machine Learning Research 10 (2009) 1571-1610 Submitted 1/08; Revised 12/08; Published 7/09

Learning Linear Ranking Functions for Beam Search with Application to Planning

Yuehua Xu XUYU@EECS.OREGONSTATE.EDU
Alan Fern AFERN@EECS.OREGONSTATE.EDU
School of Electrical Engineering and Computer Science
Oregon State University
Kelley Engineering Center
Corvallis, OR 97330

Sungwook Yoon SUNGWOOK.YOON@PARC.COM
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Editor: Michael Littman

Abstract

Beam search is commonly used to help maintain tractability in large search spaces at the expense of completeness and optimality. Here we study supervised learning of linear ranking functions for controlling beam search. The goal is to learn ranking functions that allow for beam search to perform nearly as well as unconstrained search, and hence gain computational efficiency without seriously sacrificing optimality. In this paper, we develop theoretical aspects of this learning problem and investigate the application of this framework to learning in the context of automated planning. We first study the computational complexity of the learning problem, showing that even for exponentially large search spaces the general consistency problem is in NP. We also identify tractable and intractable subclasses of the learning problem, giving insight into the problem structure. Next, we analyze the convergence of recently proposed and modified online learning algorithms, where we introduce several notions of problem margin that imply convergence for the various algorithms. Finally, we present empirical results in automated planning, where ranking functions are learned to guide beam search in a number of benchmark planning domains. The results show that our approach is often able to outperform an existing state-of-the-art planning heuristic as well as a recent approach to learning such heuristics.

Keywords: beam search, speedup learning, automated planning, structured classification

1. Introduction

Throughout artificial intelligence and computer science, heuristic search is a fundamental approach to solving complex problems. Unfortunately, when the heuristic is not accurate enough, memory and time constraints make pure heuristic search impractical. One common way to attempt to maintain tractability of heuristic search is through a pruning technique known as beam search. At each search step, beam search maintains a "beam" of the heuristically best b nodes, pruning all other nodes from the search queue. Due to this pruning, beam search is not guaranteed to be complete nor optimal. However, if the heuristic is good enough to keep a good solution path in the beam, then the solution will be found quickly.

The goal of this paper is to study the problem of learning heuristics, or ranking functions, that allow beam search to quickly find solutions, without seriously sacrificing optimality compared to unconstrained search. We consider this problem for the case of linear ranking functions, where each search node v is associated with a feature vector $f(v)$ and nodes are ranked according to $w \cdot f(v)$ where w is a weight vector. Each instance in our training set corresponds to a search space that is labeled by a set of target solutions, each solution being a (satisficing) path from the initial node to a goal node. Given a training set, our learning objective is to select a weight vector w such that a beam search of a specified beam width always maintains one of the target paths in the beam until finally reaching a goal node. Such a w effectively represents a ranking function that allows beam search to efficiently solve all of the training instances, and ideally new search problems for which the training set is representative.

Recent work (Daumé III and Marcu, 2005) has considered the problem of learning beam search ranking functions in the context of structured classification. Structured classification is the problem of learning a mapping from structured inputs (e.g., sentences) to structured outputs (e.g., syntactic parses) and there has been much recent work that extends traditional classification algorithms to this setting including conditional random fields (Lafferty et al., 2001), the generalized Perceptron algorithm (Collins, 2002), and margin optimization (Taskar et al., 2003). The approach of Daumé III and Marcu (2005) differs from prior approaches in that it explicitly views structured classification as a search problem, where given an input x, the problem of labeling x by a structured output y is treated as searching through an exponentially large set of candidate outputs. For example, in part-of-speech tagging where x is a sequence of words and y is a sequence of word tags, each node in the search space is a pair (x, y') where y' is a partial labeling of the words in x. Learning corresponds to inducing a ranking function that quickly guides the search to the search node (x, y^*) where y^* is the desired output. This framework, known as *learning as search optimization (LaSO)*, has demonstrated highly competitive performance on a number of structured classification problems.

This paper builds on the LaSO framework and makes two key contributions. First, we analyze the learning problem theoretically, in terms of its computational complexity and the convergence properties of various learning algorithms. Secondly, this paper provides an empirical evaluation in the context of automated planning, a problem that is qualitatively very different from structured classification.

Our complexity analysis considers a number of subclasses of the general beam-search learning problem. First, we provide an upper bound on the complexity of the general problem by showing that even for exponentially large search spaces, which are the norm, the consistency problem (i.e., finding a w that solves all training instances) remains in NP. Next, we identify several core tractable and intractable subclasses of the beam-search learning problem. Interestingly, some of these subclasses resemble more traditional "learning to rank" problems (Agarwal and Roth, 2005) with clear analogies to applications.

Our convergence analysis studies convergence properties of perceptron-style online learning algorithms. In prior work, Daumé III and Marcu (2005) proposed a notion of linear separability for this learning problem and proved convergence of the algorithm for linearly separable data. However, here we show that result to be inaccurate for subtle reasons and give a counter example. We then propose new notions of problem margin and show that convergence can be guaranteed for revised versions of the algorithm given positive margins. For the case where training data is ambiguous, that is, where many good solutions to a search problem are not included in the target solution set, we also give sufficient conditions on the minimum beam width to guarantee convergence. This result

also provides a formal characterization of the intuition that the learning problem should become easier as the beam width increases, by showing that the mistake bound decreases with increasing beam width.

While the LaSO framework has been empirically evaluated in structured classification, with impressive results, its utility in other types of search problems has not been demonstrated. Here we consider the application of a LaSO-style algorithm to automated planning, which is a problem that is qualitatively very different compared to structured classification. The planning problems we consider are most naturally viewed as goal-finding problems, where we must search for a short path to a goal node in an exponentially large graph. Rather, structured classification is most naturally viewed as an optimization problem, where we must search for a structured object that optimizes an objective function. While the two problem classes are related they differ in significant ways. For example, the search problems studied in structured classification typically have a single or small number of solution paths, whereas in automated planning there are often a large number of equally good solutions, which can contribute to ambiguous training data. Furthermore, the size of the search spaces encountered in automated planning are usually much larger than in structured classification, because of the larger depths and branching factors. These differences raise the empirical question of whether a LaSO-style approach will be effective in automated planning.

To evaluate this question we incorporated a LaSO-style learning mechanism into a forward state-space search planner in order to learn domain-specific heuristics, or ranking functions, from training examples. For a given planning domain, the training examples given to our learner include solution plans to a set of planning problems from the domain. The learned ranking function for a domain can then be used to guide beam search in order to solve new test problems from the same domain. We evaluate this approach on a number of benchmark planning domains and show that our learned ranking functions are often able to outperform both a state-of-the-art domain-independent planning heuristic and the heuristics learned by another recently proposed learning mechanism based on linear regression.

The remainder of this paper proceeds as follows. In Section 2, we introduce our formal setup of the beam-search learning problem and then, in Section 3, study the computational complexity of this learning problem. In Section 4, we describe two online learning mechanisms followed by their convergence analysis. In Section 5, we apply the learning problem to automated planning and present the experimental results. Finally Section 6 concludes and suggests future directions.

2. Problem Setup

In this section, we first describe two different beam search paradigms: breadth-first beam search and best-first beam search. We then introduce the learning problems that we study in these two paradigms, followed by an illustrative example from automated planning. Finally, we describe how our formulation, which was motivated by automated planning, relates to structured classification.

2.1 Beam Search

We first define breadth-first and best-first beam search, the two paradigms considered in this work. A *search space* is a tuple $\langle I, s(\cdot), f(\cdot), < \rangle$, where I is the initial search node, s is a successor function from search nodes to finite sets of search nodes, f is a feature function from search nodes to m-dimensional real-valued vectors, and $<$ is a total preference ordering on search nodes. We think of f as defining properties of search nodes that are useful for evaluating their relative goodness and $<$

as defining a canonical ordering on nodes, for example, lexicographic. In this work, we use f to define a linear ranking function $w \cdot f(v)$ on nodes where w is an m-dimensional weight vector, and nodes with larger values are considered to be higher ranked, or more preferred. Since a given w may assign two nodes the same rank, we use $<$ to break ties such that v is ranked higher than v' given $w \cdot f(v') = w \cdot f(v)$ and $v' < v$, arriving at a total rank ordering on search nodes. We denote this total rank ordering as $r(v', v|w, <)$, or just $r(v', v)$ when w and $<$ are clear from context, indicating that v is ranked higher than v'.

Given a search space $S = \langle I, s(\cdot), f(\cdot), < \rangle$, a weight vector w, and a beam width b, *breadth-first beam search* simply conducts breadth-first search, but at each search depth keeps only the b highest ranked nodes according to r. More formally, breadth-first beam search generates a unique *beam trajectory* $(B_0, B_1, \ldots)$ as follows,

- $B_0 = \{I\}$ is the initial beam;
- $C_{j+1} = \textbf{BreadthExpand}(B_j, s(\cdot)) = \bigcup_{v \in B_j} s(v)$ is the depth $j+1$ *candidate set* of the depth j beam;
- B_{j+1} is the unique set of b highest ranked nodes in C_{j+1} according to the total ordering r.

Note that for any j, $|C_j| \leq cb$ and $|B_j| \leq b$, where c is the maximum number of children of any search node.

Best-first beam search is almost identical to breadth-first beam search except that we replace the function **BreadthExpand** with $\textbf{BestExpand}(B_j, s(\cdot)) = B_j \cup s(v^*) - v^*$, where v^* is the unique highest ranking node in B_j. Thus, instead of expanding all nodes in the beam at each search step, best-first search is more conservative and only expands the single best node. Note that unlike breadth-first search this can result in beams that contain search nodes from different depths of the search space relative to I.

2.2 Learning Problems

Our learning problems provide training sets of pairs $\{\langle S_i, P_i \rangle\}$, where the $S_i = \langle I_i, s_i(\cdot), f_i(\cdot), <_i \rangle$ are search spaces constrained such that each f_i has the same dimension. As described in more detail below, the P_i encode sets of *target search paths* that describe desirable search paths through the corresponding search spaces. Roughly speaking the learning goal is to learn a ranking function that can produce a beam trajectory of a specified width for each search space that contains at least one of the corresponding target paths in the training data. For example, in the context of automated planning, the S_i would correspond to planning problems from a particular domain, encoding the state space and available actions, and the P_i would encode optimal or satisficing plans for those problems. A successfully learned ranking function would be able to quickly find at least one of the target solution plans for each training problem and ideally new target problems.

We represent each set of target search paths as a sequence $P_i = (P_{i,0}, P_{i,1}, \ldots, P_{i,d})$ of sets of search nodes where $P_{i,j}$ contains target nodes at depth j and $P_{i,0} = \{I_i\}$. It is useful to think about $P_{i,d}$ as encoding the *goal nodes* of the i'th search space. We will refer to the maximum size t of any target node set $P_{i,j}$ as the *target width* of P_i, which will be referred to in our complexity analysis. The generality of this representation for target paths allows for pathological targets where certain nodes do not lead to the goal. In order to arrive at convergence results, we rule out such possibilities by assuming that the training set is *dead-end free*. That is, for all i and $j < d$ each $v \in P_{i,j}$ has at

least one child node $v' \in P_{i,j+1}$. Note that in almost all real problems this property will be naturally satisfied. For our complexity analysis, we will not need to assume any special properties of the target search paths P_i.

Intuitively, for a dead-end free training set, each P_i represents a layered directed graph with at least one path from each target node to a goal node in $P_{i,d}$. Thus, the training set specifies not only a set of goals for each search space but also gives possible solution paths to the goals. For simplicity, we assume that all target solution paths have depth d, but all results easily generalize to non-uniform depths.

For breadth-first beam search we specify a learning problem by giving a training set and a beam width $\langle\{\langle S_i, P_i\rangle\}, b\rangle$. The objective is to find a weight vector w that generates a beam trajectory containing at least one of the target paths for each training instance. More formally, we are interested in the consistency problem:

Definition 1 (Breadth-First Consistency) *Given the input $\langle\{\langle S_i, P_i\rangle\}, b\rangle$ where b is a positive integer and $P_i = (P_{i,0}, P_{i,1}, \ldots, P_{i,d})$, the breadth-first consistency problem asks us to decide whether there exists a weight vector w such that for each S_i, the corresponding beam trajectory $(B_{i,0}, B_{i,1}, \ldots, B_{i,d})$, produced using w with a beam width of b, satisfies $B_{i,j} \cap P_{i,j} \neq \emptyset$ for each j?*

A weight vector that demonstrates a "yes" answer is guaranteed to allow a breath-first beam search of width b to uncover at least one goal node (i.e., a node in $P_{i,d}$) within d beam expansions for all training instances.

Unlike the case of breadth-first beam search, the length of the beam trajectory required by best-first beam search to reach a goal node can be greater than the depth d of the target paths. This is because best-first beam search, does not necessarily increase the maximum depth of search nodes in the beam at each search step. Thus, in addition to specifying a beam width for the learning problem, we also specify a maximum number of search steps, or horizon, h. The objective is to find a weight vector that allows a best-first beam search to find a goal node within h search steps, while always keeping some node from the target paths in the beam.

Definition 2 (Best-First Consistency) *Given the input $\langle\{\langle S_i, P_i\rangle\}, b, h\rangle$, where b and h are positive integers and $P_i = (P_{i,0}, \ldots, P_{i,d})$, the best-first consistency problem asks us to decide whether there is a weight vector w that produces for each S_i a beam trajectory $(B_{i,0}, \ldots, B_{i,k})$ of beam width b, such that $k \leq h$, $B_{i,k} \cap P_{i,d} \neq \emptyset$ (i.e., $B_{i,k}$ contains a goal node), and each $B_{i,j}$ for $j < k$ contains at least one node in $\bigcup_j P_{i,j}$?*

Again, a weight vector that demonstrates a "yes" answer is guaranteed to allow a best-first beam search of width b to find a goal node in h search steps for all training instances.

2.2.1 EXAMPLE FROM AUTOMATED PLANNING.

Figure 1, shows a pictorial example of a single training example from an automated planning problem. The planning domain in this example is Blocksworld where individual problems involve transforming an initial configuration of blocks to a goal configuration using simple actions such as picking up, putting down, and stacking the various blocks. The figure shows a search space S_i where each node corresponds to a configuration of blocks and the arcs indicate when it is possible to take an action that transitions from one configuration to another. The figure depicts, via highlighted nodes,

two target paths. The label P_i would encode these target paths by a sequence $P_i = (P_{i,0}, P_{i,1}, \ldots, P_{i,4})$ where $P_{i,j}$ contains the set of all highlighted target nodes at depth j. A solution weight vector, for this training example, would be required to keep at least one of the highlighted paths in the beam until uncovering the goal node.

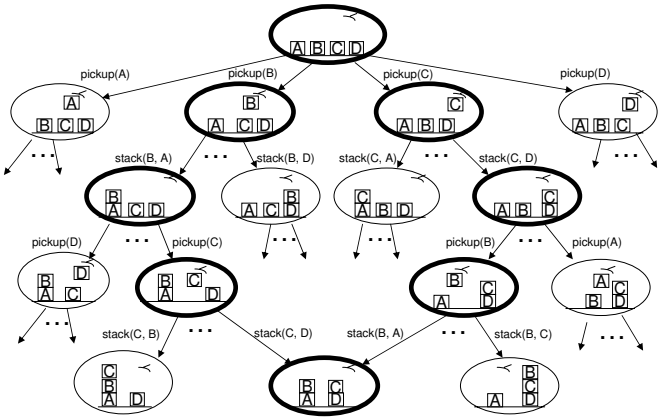


Figure 1: An example from automated planning.

2.2.2 EXAMPLE FROM STRUCTURED CLASSIFICATION

Daumé III and Marcu (2005) considered learning ranking functions to control beam search in the context of structured classification. Structured classification involves learning a function that maps structured inputs x to structured outputs y. As an example, consider part-of-speech tagging where the inputs correspond to English sentences and the correct output for a sentence is the sequence of part-of-speech tags for the words in the sentence. Figure 2 shows how Daumé III and Marcu (2005) formulated a single instance of part-of-speech tagging as a search problem. Each search node is a pair (x, y') where x is the input sentence and y' is a partial labeling of the words in x by part-of-speech tags. The arcs in this space correspond to search steps that label words in the sentence in a left-to-right order by extending y' in all possible ways by one element. The leaves, or terminal nodes, of this space correspond to all possible complete labelings of x. Given a ranking function and a beam width, Daumé III and Marcu (2005) return a predicted output for x by conducting a beam search until a terminal node becomes the highest ranked node in the beam, and then return the output component of that terminal node. This approach to making predictions suggests that the learning objective should require that we learn a ranking function such that the goal terminal node, is the first terminal node to become highest ranked in the beam. In the figure, there is a single goal terminal node (x, y) where y is the correct labeling of x and there is a unique target path to this goal.

From the above example, we see that there is a difference between the learning objective used by Daumé III and Marcu (2005) for structured classification and the learning objective under our formulation, which was motivated by automated planning. In particular, our formulation does not force the goal node to be the highest ranked node in the final beam, but rather only requires that a goal node appear somewhere in the final beam. While these formulations appear quite different, it turns out that they are polynomially reducible to one another, which we prove in Appendix A.

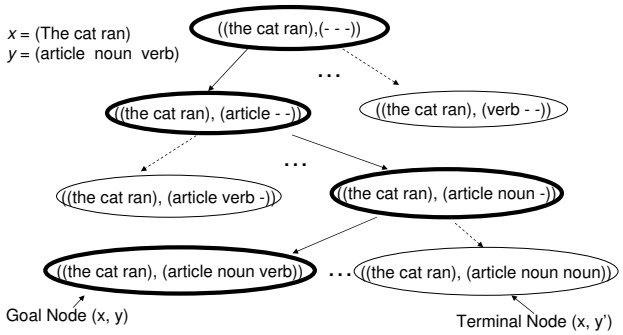


Figure 2: An example from structured classification.

Thus, all of the results in this paper apply equally well to the structured-classification formulation of Daumé III and Marcu (2005).

3. Computational Complexity

In this section, we study the computational complexity of the above consistency problems. We first focus on breadth-first beam search, and then give the corresponding best-first results at the end of this section. It is important to note that the size of the search spaces will typically be exponential in the encoding size of the learning problem. For example, in automated planning, standard languages such as PDDL (McDermott, 1998) are used to compactly encode planning problems that are potentially exponentially large, in terms of the number of states, with respect to the PDDL encoding size. Throughout this section we measure complexity in terms of the problem encoding size, not the potentially exponentially larger search space size. All discussions in this section apply to general search spaces and are not tied to a particular language for describing search space such as PDDL.

Our complexity analysis will consider various sub-classes of the breadth-first consistency problem, where the sub-classes will be defined by placing constraints on the following problem parameters: n - the number of training instances, d - the depth of target solution paths, c - the maximum number of children of any search node, t - the maximum target width of any P_i as defined in Section 2.2, and b - the beam width. Figure 3 gives a pictorial depiction of these key problem parameters. Throughout the complexity analysis we will restrict our attention to problem classes where the maximum number of children c and beam width b are polynomial in the problem size, which are necessary conditions to ensure that each beam search step requires only polynomial time and space. We will also assume that all feature functions can be evaluated in polynomial time in the problem size.

Note that restricting the number of children c may rule out the use of certain search space encodings for some problems. For example, in a multi-agent planning scenario, there are an exponential number of joint actions to consider from each state, and thus an exponential number of children. However, here it is possible to re-encode the search space by increasing the depth of the search tree, so that each joint action is encoded by a sequence of steps where each agent selects an action in turn followed by all of them executing the selected actions. The resulting search space has only a polynomial number of children and thus satisfies our assumption, though the required search depth

has increased. This form of re-encoding from a search space with exponentially many children to one with polynomially many children can be done whenever the actions in the original space have a compact, factored encoding, which is typically the case in practice.

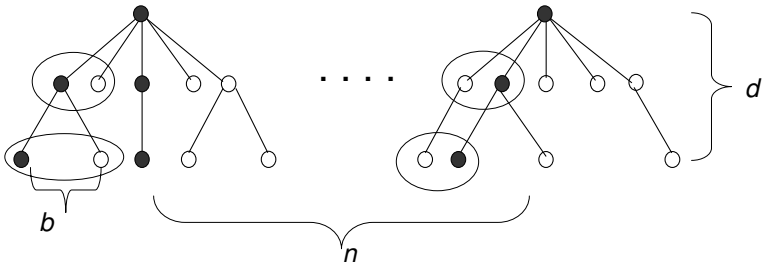


Figure 3: The key problem parameters: n - the number of training instances, d - the depth of target solution paths, b - the beam width. Not depicted in the figure are: c - maximum number of children of any node, t - the maximum target width of any example.

3.1 Hardness Upper Bounds

We first show an upper bound on the complexity of breadth-first consistency by proving that the general problem is in NP even for exponentially large search spaces.

Observe that given a weight vector w and beam width b, we can easily generate a unique depth d beam trajectory for each training instance. Our upper bound is based on considering the inverse problem of checking whether a set of hypothesized beam trajectories, one for each training instance, could have been generated by some weight vector. The algorithm *TestTrajectories* in Figure 4 efficiently carries out this check. The main idea is to observe that for any search space S it is possible to efficiently check whether there is a weight vector that starting with a beam B could generate a beam B' after one step of breadth-first beam search. This can be done by constructing an appropriate set of linear constraints on the weight vector w that are required to generate B' from B. In particular, we first generate the set of candidate nodes C from B by unioning all children of nodes in B. Clearly we must have $B' \subseteq C$ in order for there to be a solution weight vector. If this is the case then we create a linear constraint for each pair of nodes (u,v) such that $u \in B'$ and $v \in C - B'$, which forces u to be preferred to v:

$$w \cdot f(u) > w \cdot f(v)$$

where $w = (w_1, w_2, \ldots, w_m)$ are the constraint variables and $f(\cdot) = (f_1(\cdot), f_2(\cdot), \ldots, f_m(\cdot))$ is the vector of feature functions. Note that if u is more preferred than v in the total preference ordering, then we only need to require that $w \cdot f(u) \geq w \cdot f(v)$. The overall algorithm *TestTrajectories* simply creates this set of constraints for each consecutive pair of beams in each beam trajectory and then tests to see whether there is a w that satisfies all of the constraints.

Lemma 3 *Given a set of search spaces $\{S_i\}$ and a corresponding set of width b beam trajectories $\{(B_{i,0}, \ldots, B_{i,d})\}$, the algorithm TestTrajectories (Figure 4) decides in polynomial time whether there exists a weight vector w that can generate $(B_{i,0}, \ldots, B_{i,d})$ in S_i for all i.*

Proof It is straightforward to show that w satisfies the constraints generated by *TestTrajectories* iff for each i, j, $r(v', v| <_i, w)$ leads beam search to generate $B_{i,j+1}$ from $B_{i,j}$. The linear program contains m variables and at most $ndcb^2$ constraints. Since we are assuming that the maximum number of children of a node v is polynomial in the size of the learning problem, the size of the linear program is also polynomial and thus can be solved in polynomial time (Khachiyan, 1979). ■

This lemma shows that sets of beam trajectories can be used as efficiently-checkable certificates for breadth-first consistency, which leads to an upper bound on the problem's complexity.

Theorem 4 *Breadth-first consistency is in NP.*

Proof Given a learning problem $\langle \{\langle S_i, P_i \rangle\}, b \rangle$ our certificates correspond to sets of beam trajectories $\{(B_{i,0}, \ldots, B_{i,d})\}$ each of size at most $O(ndb)$ which is polynomial in the problem size. The certificate can then be checked in polynomial time to see if for each i, $(B_{i,0}, \ldots, B_{i,d})$ contains a target solution path encoded in P_i as required by Definition 1. If it is consistent then according to Lemma 3 we can efficiently decide whether there is a w that can generate $\{(B_{i,0}, \ldots, B_{i,d})\}$. ■

This result suggests an enumeration-based decision procedure for breadth-first consistency as given in Figure 4. In that procedure, the function **Enumerate** creates a list of all possible combinations of beam trajectories for the training data. Thus, each element of this list is a list of beam trajectories, one for each training example, where a beam trajectory is simply a sequence of sets of nodes that are selected from the given search space. For each enumerated combination of beam trajectories, the function **IsConsistent** checks whether the beam trajectory for each example contains a target path for that example and if so *TestTrajectories* will be called to determine whether there exists a weight vector that could produce those trajectories. The following gives us the worst case complexity of this algorithm in terms of the key problem parameters.

Theorem 5 *The procedure ExhaustiveAlgorithm (Figure 4) decides breadth-first consistency and returns a solution weight vector if there is a solution in time* $O\left((t + poly(m))\,(cb)^{bdn}\right)$.

Proof We first bound the number of certificates. Breadth-first beam search expands nodes in the current beam, resulting in at most cb nodes, from which b nodes are selected for the next beam. Enumerating these possible choices over d levels and n trajectories, one for each training instance, we can bound the number of certificates by $O\left((cb)^{bdn}\right)$. For each certificate the enumeration process checks consistency with the target paths $\{P_i\}$ in time $O(tbdn)$ and then calls *TestTrajectories* which runs in time $\text{poly}(m, ndcb^2)$. The total time complexity then is $O\left(\left(tbdn + \text{poly}(m, ndcb^2)\right)(cb)^{bdn}\right)$ $= O\left((t + \text{poly}(m))\,(cb)^{bdn}\right)$. ■

Not surprisingly the complexity is exponential in the beam width b, target path depth d, and number of training instances n. However, it is polynomial in the maximum number of children c and the maximum target width t. Thus, breadth-first consistency can be solved in polynomial time for any problem class where b, d, and n are constants. Of course, for most problems these constants would be too large for this result to be of practical interest. This leads to the question of whether we can do better than the exhaustive algorithm for restricted problem classes. For at least one problem class we can.

```
ExhaustiveAlgorithm ({⟨S_i, P_i⟩}, b)
Γ = Enumerate({⟨S_i, P_i⟩}, b)
// enumerates all possible sets of beam trajectories
for each {(B_{i,0} ..., B_{i,d})} ∈ Γ
    if IsConsistent({P_i}, {(B_{i,0} ..., B_{i,d})}) then
        w= TestTrajectories({S_i}, {(B_{i,0}, ..., B_{i,d})})
        if w ≠ false then
            return w
return false
```

```
TestTrajectories({S_i}, {(B_{i,0}, ..., B_{i,d})})
// S_i = ⟨I_i, s_i(·), f_i(·), <_i⟩
construct a linear programming problem LP as below
    the variables are w = {w_1, w_2, ..., w_m}
    for (i, j) ∈ {1, ..., n} × {1, ..., d}
        C_{i,j} =BreadthExpand(B_{i,j-1}, s_i(·))
        if B_{i,j} ⊆ C_{i,j} then
            for each u ∈ B_{i,j} and v ∈ C_{i,j} − B_{i,j}
                if v <_i u then
                    add a constraint w · f_i(u) ≥ w · f_i(v)
                else add a constraint w · f_i(u) > w · f_i(v)
        else return false
w = LPSolver(LP)
if LP is solved then
    return w
return false
```

Figure 4: The exhaustive algorithm for breadth-first consistency.

Theorem 6 *The class of breadth-first consistency problems where $b = 1$ and $t = 1$ is solvable in polynomial time.*

Proof Given a learning problem $\langle \{\langle S_i, P_i \rangle\}, b \rangle$ where $P_i = (P_{i,0}, \ldots, P_{i,d})$, $t = 1$ implies that each $P_{i,j}$ contains exactly one node. Since the beam width $b = 1$, then the only way that a beam trajectory $(B_{i,0}, \ldots, B_{i,d})$ can satisfy the condition $B_{i,j} \cap P_{i,j} \neq \emptyset$ for any i, j, is for $B_{i,j} = P_{i,j}$. Thus there is exactly one beam trajectory for each training example, equal to the target trajectory, and using Lemma 3 we can check for a solution weight vector for these trajectories in polynomial time. ■

This problem class, as depicted in Figure 5, corresponds to the case where each training instance is labeled by exactly a single solution path and we are asked to find a w that leads a greedy hill-climbing search, or reactive policy, to follow those paths. This is a common learning setting, for example, when attempting to learn reactive control policies based on demonstrations of target policies, perhaps from an expert, as in Khardon (1999).

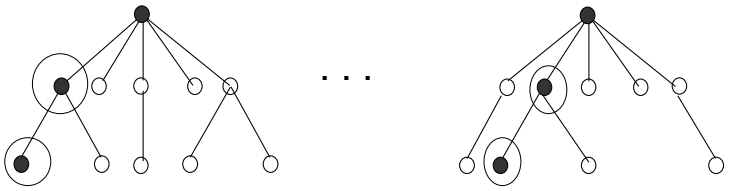


Figure 5: A tractable class of breadth-first consistency, where $b = 1$ and $t = 1$.

3.2 Hardness Lower Bounds

Unfortunately, outside of the above problem classes it appears that breadth-first consistency is computationally hard even under strict restrictions. In particular, the following three results show that if any one of b, d, or n are not bounded then the consistency problem is hard even when the other problem parameters are small constants.

First, we show that the problem class where $n = d = t = 1$ but $b \geq 1$ is NP-complete. That is, a single training instance involving a depth one search space is sufficient for hardness. This problem class, resembles more traditional ranking problems and has a nice analogy in the application domain of web-page ranking, where the depth 1 leaves of our search space correspond to possibly relevant web-pages for a particular query. One of those pages is marked as a target page, for example, the page that a user eventually went to. The learning problem is then to find a weight vector that will cause for the target page to be ranked among the top b pages. Our result shows that this problem is NP-complete and hence will be exponential in b unless $P = NP$.

Theorem 7 *The class of breadth-first consistency problems where $n = 1$, $d = 1$, $t = 1$, and $b \geq 1$ is NP-complete.*

Proof Our reduction is from the Minimum Disagreement problem for linear binary classifiers, which was proven to be NP-complete by Hoffgen et al. (1995). The input to this problem is a training set $T = \{x_1^+, \cdots, x_{r_1}^+, x_1^-, \cdots, x_{r_2}^-\}$ of positive and negative m-dimensional vectors and a positive integer k. A weight vector w classifies a vector as positive iff $w \cdot x \geq 0$ and otherwise as negative. The Minimum Disagreement problem is to decide whether there exists a weight vector that commits no more than k misclassification.

Given a Minimum Disagreement problem we construct an instance $\langle \langle S_1, P_1 \rangle, b \rangle$ of the breadth-first consistency problem as follows. Assume without loss of generality $S_1 = \langle I, s(\cdot), f(\cdot), < \rangle$. Let $s(I) = \{q_0, q_1, \cdots, q_{r_1+r_2}\}$. For each $i \in \{1, \cdots, r_1\}$, define $f(q_i) = -x_i^+ \in R^m$. For each $i \in \{1, \cdots, r_2\}$,define $f(q_{i+r_1}) = x_i^- \in R^m$. Define $f(q_0) = 0 \in R^m$, $P_1 = (\{I\}, \{q_0\})$ and $b = k+1$. Define the total ordering $<$ to be a total ordering in which $q_i < q_0$ for every $i = 1, \ldots, r_1$ and $q_0 < q_i$ for every $i = r_1 + 1, \ldots, r_1 + r_2$.We claim that there exists a linear classifier with at most k misclassifications if and only if there exists a solution to the corresponding consistency problem.

First, suppose there exists a linear classifier $w \cdot x \geq 0$ with at most k misclassifications. Using the weight vector w, we have

- $w \cdot f(q_0) = 0$;

- for $i = 1, \cdots, r_1$:
  if $w \cdot x_i^+ \geq 0, w \cdot f(q_i) = w \cdot (-x_i^+) \leq 0$;
  if $w \cdot x_i^+ < 0, w \cdot f(q_i) = w \cdot (-x_i^+) > 0$;

- for $i = r_1 + 1, \ldots, r_1 + r_2$:
  if $w \cdot x_i^- \geq 0, w \cdot f(q_i) = w \cdot x_i^- \geq 0$;
  if $w \cdot x_i^- < 0, w \cdot f(q_i) = w \cdot x_i^- < 0$.

For $i = 1, \cdots, r_1 + r_2$, the node q_i in the consistency problem is ranked lower than q_0 if and only if its corresponding example in the Minimum Disagreement problem is labeled correctly, is ranked higher than q_0 if and only if its corresponding example in the Minimum Disagreement problem is labeled incorrectly. Therefore, there are at most k nodes which are ranked higher than q_0. With beam width $b = k + 1$, the beam $B_{i,1}$ is guaranteed to contain node q_0, indicating that w is a solution to the consistency problem.

On the other hand, suppose there exists a solution w to the consistency problem. There are at most $b - 1 = k$ nodes that are ranked higher than q_0. That is, at least $r_1 + r_2 - k$ nodes are ranked lower than q_0. For $i = 1, \ldots, r_1$, q_i is ranked lower than q_0 if and only if $w \cdot f(q_i) \leq w \cdot f(q_0)$. For $i = r_1 + 1, \ldots, r_1 + r_2$, q_i is ranked lower than q_0 if and only if $w \cdot f(q_i) < w \cdot f(q_0)$. Since $w \cdot f(q_0) = 0$, we have

- for $i = 1, \cdots, r_1$:
  $w \cdot f(q_i) \leq 0 \Rightarrow w \cdot (-x_i^+) \leq 0 \Rightarrow w \cdot x_i^+ \geq 0$;

- for $i = r_1 + 1, \ldots, r_1 + r_2$:
  $w \cdot f(q_i) < 0 \Rightarrow w \cdot x_i^- < 0 \Rightarrow w \cdot x_i^- < 0$.

Therefore, using the linear classifier $w \cdot x \geq 0$, at least $r_1 + r_2 - k$ nodes are labeled correctly, that is, it makes at most k misclassifications.

Since the time required to construct the instance $\langle \langle S_1, P_1 \rangle, b \rangle$ from T, k is polynomial in the size of T, k, we conclude that the consistency problem is NP-Complete even restricted to $n = 1$, $d = 1$ and $t = 1$. ■

The next result shows that if we do not bound the number of training instances n, then the problem remains hard even when the target path depth and beam width are equal to one. Interestingly, this subclass of breadth-first consistency corresponds to the multi-label learning problem as defined in Jin and Ghahramani (2002). In multi-label learning each training instance can be viewed as a bag of m-dimensional vectors, some of which are labeled as positive, which in our context correspond to the target nodes. The learning goal is to find a w that for each bag, ranks one of the positive vectors as best.

Theorem 8 *The class of breadth-first consistency problems where $d = 1$, $b = 1$, $c = 6$, $t = 3$, and $n \geq 1$ is NP-complete.*

Proof The proof is by reduction from 3-SAT (Garey and Johnson, 1979), which is the following.

"Given a set U of boolean variables, a collection Q of clauses over U such that each clause $q \in Q$ has $|q| = 3$, decide whether there a satisfying truth assignment for C."

Let $U = \{u_1, \ldots, u_m\}$, $Q = \{q_{11} \vee q_{12} \vee q_{13}, \ldots, q_{n1} \vee q_{n2} \vee q_{n3}\}$ be an instance of the 3-SAT problem. Here, $q_{ij} = u$ or $\neg u$ for some $u \in U$. We construct from U, Q an instance $\langle \{\langle S_i, P_i \rangle\}, b \rangle$ of the breadth-first consistency problem as follows. For each clause $q_{i1} \vee q_{i2} \vee q_{i3}$, let $s_i(I_i) = \{p_{i,1}, \cdots, p_{i,6}\}$, $P_i = (\{I_i\}, \{p_{i,1}, p_{i,2}, p_{i,3}\})$, $b = 1$, and the total ordering $<_i$ is defined so that $p_{i,j} <_i p_{i,k}$ for $j = 1,2,3$ and $k = 4,5,6$. Let $e_k \in \{0,1\}^m$ denote a vector of zeros except a 1 in the k'th component. For each $i = 1, \ldots, n$, $j = 1,2,3$, if $q_{ij} = u_k$ for some k then $f_i(p_{i,j}) = e_k$ and $f_i(p_{i,j+3}) = -e_k/2$, otherwise if $q_{ij} = \neg u_k$ for some k then $f_i(p_{i,j}) = -e_k$ and $f_i(p_{i,j+3}) = e_k/2$. We claim that there exists a satisfying truth assignment if and only if there exists a solution to the corresponding consistency problem.

First, suppose that there exists a satisfying truth assignment. Let $w = (w_1, \cdots, w_m)$, where $w_k = 1$ if u_k is true, and $w_k = -1$ if u_k is false in the truth assignment. For each $i = 1, \ldots, n$, $j = 1, \ldots, 3$, we have:

- if q_{ij} is true, then
  $w \cdot f_i(p_{i,j}) = 1$ and $w \cdot f_i(p_{i,j+3}) = -1/2$;
- if q_{ij} is false, then
  $w \cdot f_i(p_{i,j}) = -1$ and $w \cdot f_i(p_{i,j+3}) = 1/2$.

Note that for each clause $q_{i1} \vee q_{i2} \vee q_{i3}$, at least one of the literals is true. Thus, for every set of nodes $\{p_{i,1}, p_{i,2}, p_{i,3}\}$, at least one of the nodes will have the highest rank value equal to 1, resulting in $B_{i,1} = \{v\}$ where $v \in \{p_{i,1}, p_{i,2}, p_{i,3}\}$. By the definition, the weight vector w is a solution to the consistency problem.

On the other hand, suppose that there exists a solution $w = (w_1, \ldots, w_m)$ to the consistency problem. Assume the beam trajectory for each i is $(\{I_i\}, \{v_i\})$. Then $v_i = p_{i,j}$ for some $j \in \{1,2,3\}$, and for this i and j, $q_{ij} = u_k$ or $\neg u_k$ for some k. Let u_k be true if $q_{ij} = u_k$ and be false if $q_{ij} = \neg u_k$. As long as there is no contradiction in this assignment, this is a satisfying truth assignment because at least one of $\{q_{i1}, q_{i2}, q_{i3}\}$ is true for every i, that is, every clause is true.

Now we will prove that there is no contradiction in this assignment, that is, any variable is assigned either true or false, but not both. Note that for any node $v \in \{p_{i,1}, p_{i,2}, p_{i,3}\}$, there always exists a node $v' \in \{p_{i,4}, \ldots, p_{i,6}\}$ such that:

- $w \cdot f_i(v) < 0 \Leftrightarrow w \cdot f_i(v') > 0$;
- $w \cdot f_i(v) > 0 \Leftrightarrow w \cdot f_i(v') < 0$;
- $w \cdot f_i(v) = 0 \Leftrightarrow w \cdot f_i(v') = 0$.

Then because of the total ordering $<_i$ we defined, the node $v_i = p_{i,j}$ appearing in the beam trajectory, must has $w \cdot f_i(v_i) > 0$. Assume without loss of generality that $q_{ij} = u_k$, then u_k is assigned to be true. Although $\neg u_k$ might appear in other clauses, for example, $q_{i'j'} = \neg u_k$, its corresponding node $p_{i',j'}$ can never appear in the beam trajectory because $w \cdot f_{i'}(p_{i',j'}) = w \cdot (-e_k) = -w \cdot e_k = -w \cdot f_i(p_{i,j}) < 0$. Therefore, u_k will never be assigned false. A similar proof can be given for the case of $q_{ij} = \neg u_k$.

Since the time required to construct the instance $\langle \{\langle S_i, P_i \rangle\}, b \rangle$ from U, Q is polynomial in the size of U, Q, we conclude that the consistency problem is NP-Complete for the case of $d = 1, b = 1$, $c = 6$ and $t = 3$.

■

Finally, we show that when the depth d is unbounded the consistency problem remains hard even when $b = n = 1$.

Theorem 9 *The class of breadth-first consistency problems where $n = 1$, $b = 1$, $c = 6$, $t = 3$, and $d \geq 1$ is NP-complete.*

Proof Assume $x = \langle \{\langle S_i, P_i \rangle | i = 1, \ldots, n\}, b \rangle$, where $S_i = \langle I_i, s_i(\cdot), f_i(\cdot), <_i \rangle$ and $P_i = (\{I_i\}, P_{i,1})$, is an instance of the consistency problem with $d = 1$, $b = 1$, $c = 6$ and $t = 3$. We can construct an instance y of the consistency problem with $n = 1, b = 1, c = 6$, and $t = 3$. Let $y = \langle \langle \bar{S}_1, \bar{P}_1 \rangle, b \rangle$ where $\bar{S}_1 = \langle I_1, \bar{s}(\cdot), \bar{f}(\cdot), \bar{<} \rangle$, and $\bar{P}_1 = (\{I_1\}, P_{1,1}, P_{2,1}, \ldots, P_{t,1})$. We define $\bar{s}(\cdot), \bar{f}(\cdot), \bar{<}$ as below.

- $\bar{s}(I_1) = s_1(I_1), \bar{f}(I_1) = f_1(I_1)$;
- for each $i = 1, \ldots, n-1$
  $\forall v \in s_i(I_i), \bar{f}(v) = f_i(v)$ and $\bar{s}(v) = s_{i+1}(I_{i+1})$;
  $\forall (v, v') \in s_i(I_i), \bar{<}(v, v') =<_i (v, v')$;
- $\forall v \in s_n(I_n), \bar{f}(v) = f_n(v)$;
  $\forall (v, v') \in s_n(I_n), \bar{<}(v, v') =<_n (v, v')$.

Obviously, a weight vector w is a solution for the instance x if and only if w is a solution for the constructed instance y. ■

b	n	d	c	t	Complexity
poly	≥ 1	≥ 1	poly	≥ 1	NP
K	K	K	poly	≥ 1	P
1	≥ 1	≥ 1	poly	1	P
poly	1	1	poly	1	NP-Complete
1	≥ 1	1	6	3	NP-Complete
1	1	≥ 1	6	3	NP-Complete

Figure 6: Complexity results for breadth-first consistency. Each row corresponds to a sub-class of the problem and indicates the computational complexity. K indicates a constant value and "poly" indicates that the quantity must be polynomially related to the problem size.

Figure 6 summarizes our main complexity results from this section for breadth-first consistency. For best-first beam search, most of these results can be carried over. Recall that for best-first consistency the problem specifies a search horizon h in addition to a beam width. Using a similar approach as above we can show that best-first consistency is in NP assuming that h is polynomial in the problem size, which is a reasonable assumption. Similarly, one can extend the polynomial time result for fixed b, n, and d. The remaining results in the table can be directly transferred to best-first search, since in each case either $b = 1$ or $d = 1$ and best-first beam search is equivalent to breadth-first beam search in either of these cases.

4. Convergence of Online Updates

In the previous section, we identified a limited set of tractable problem classes and saw that even very restricted classes remain NP-hard. We also saw that some of these hard classes had interesting application relevance. Thus, it is desirable to consider efficient learning mechanisms that work well in practice. Below we describe two such algorithms based on online perceptron updates.

4.1 Online Perceptron Updates

Figure 7 gives the LaSO-BR algorithm for learning ranking functions for breadth-first beam search. It resembles the *learning as search optimization (LaSO)* algorithm for best-first search by Daumé III and Marcu (2005). LaSO-BR iterates through all training instances $\langle S_i, P_i \rangle$ and for each one conducts a beam search of the specified width. After generating the depth j beam for the ith training instance, if at least one of the target nodes in $P_{i,j}$ are in the beam then no weight update occurs. Rather, if none of the target nodes in $P_{i,j}$ are in the beam then a search error is flagged and weights are updated according to the following perceptron-style rule,

$$w = w + \alpha \cdot \left(\frac{\sum_{v^* \in P_{i,j} \cap C} f(v^*)}{|P_{i,j} \cap C|} - \frac{\sum_{v \in B} f(v)}{b} \right)$$

where $0 < \alpha \leq 1$ is a learning rate parameter, B is the current beam and C is the candidate set from which B was generated (i.e., the beam expansion of the previous beam). For simplicity of notation, here we assume that f is a feature function for all training instances. Intuitively this weight update moves the weights in the direction of the average feature function of target nodes that appear in C, and away from the average feature function of non-target nodes in the beam. This has the effect of increasing the rank of target nodes in C and decreasing the rank of non-targets in the beam. Ideally, this will cause at least one of the target nodes to become preferred enough to remain on the beam next time through the search. Note that the use of averages over target and non-target nodes is important so as to account for the different sizes of these sets of nodes. After each weight update, the beam is reset to contain only the set of target nodes in C and the beam search then continues. Importantly, on each iteration, the processing of each training instance is guaranteed to terminate in d search steps.

Figure 8 gives the LaSO-BST algorithm for learning in best-first beam search, which is a slight modification of the original LaSO algorithm. The main difference compared to the original LaSO is in the weight update equation, a change that appears necessary for our convergence analysis. The algorithm is similar to LaSO-BR except that a best-first beam search is conducted, which means that termination for each training instance is not guaranteed to be within d steps. Rather, the number of search steps for a single training instance remains unbounded without further assumptions, which we will address later in this section. In particular, there is no bound on the number of search steps between weight updates for a given training example. This difference between LaSO-BR and LaSO-BST was of great practical importance in our automated planning application. In particular, LaSO-BST typically did not produce useful learning results due to the fact that the number of search steps between weight updates was extremely large. Note that in the case of structured classification, Daumé III and Marcu (2005) did not experience this difficulty due to the bounded-depth nature of their search spaces.

LaSO-BR $(\{\langle S_i, P_i \rangle\}, b)$
$w \leftarrow 0$
repeat until w is unchanged **or** a large number of iterations
  for every i
    Update-BR(S_i, P_i, b, w)
return w

Update-BR (S_i, P_i, b, w)
// $S_i = \langle I_i, s_i(\cdot), f(\cdot), <_i \rangle$ and $P_i = (P_{i,0}, \ldots, P_{i,d})$
$B \leftarrow \{I_i\}$ // initial beam
for $j = 1, \ldots, d$
  $C \leftarrow$ **BreadthExpand**$(B, s_i(\cdot))$
  for every $v \in C$
    $H(v) \leftarrow w \cdot f(v)$ // compute heuristic value of v
  Order C according to H and the total ordering $<_i$
  $B \leftarrow$ the first b nodes in C
  if $B \cap P_{i,j} = \emptyset$ **then**
    $w \leftarrow w + \alpha \cdot \left(\frac{\sum_{v^* \in P_{i,j} \cap C} f(v^*)}{|P_{i,j} \cap C|} - \frac{\sum_{v \in B} f(v)}{b} \right)$
    $B \leftarrow P_{i,j} \cap C$
return

Figure 7: The LaSO-BR online algorithm for breadth-first beam search.

4.2 Previous Result and Counter Example

Adjusting to our terminology, Daumé III and Marcu (2005) defined a training set to be *linear separable* iff there is a weight vector that solves the corresponding consistency problem. Also for linearly separable data they defined a notion of margin of a weight vector, which we refer to here as the *search margin*. The formal definition of search margin is given below.

Definition 10 (Search Margin) *The search margin of a weight vector w for a linearly separable training set is defined as $\gamma = min_{\{(v^*,v)\}}(w \cdot f(v^*) - w \cdot f(v))$, where the set $\{(v^*, v)\}$ contains any pair where v^* is a target node and v is a non-target node that was compared during the beam search guided by w.*

Daumé III and Marcu (2005) state that the existence of a w with positive search margin, which implies linear separability, implies convergence of the original LaSO algorithm after a finite number of weight updates. On further investigation, we have found that a positive search margin is not sufficient to guarantee convergence for LaSO, LaSO-BR, or LaSO-BST. Intuitively, the key difficulty is that our learning problem contains hidden state in the form of the desired beam trajectory. Given the beam trajectory of a consistent weight vector one can compute the weights, and likewise given consistent weights one can compute the beam trajectory. However, we are given neither to begin with and our approach can be viewed as an online EM-style algorithm, which alternates between

LaSO-BST $(\{\langle S_i, P_i\rangle\}, b)$
$w \leftarrow 0$
repeat until w is unchanged **or** a large number of iterations
  for every i
    Update-BST(S_i, P_i, b, w)
return w

Update-BST (S_i, P_i, b, w)
// $S_i = \langle I_i, s_i(\cdot), f(\cdot), <_i\rangle$ and $P_i = (P_{i,0}, \ldots, P_{i,d})$
$B \leftarrow \{I_i\}$ // initial beam
$\bar{P} = P_{i,0} \cup P_{i,2} \cup \ldots \cup P_{i,d}$
while $B \cap P_{i,d} = \emptyset$
  $C \leftarrow$ **BestExpand**$(B, s_i(\cdot))$
  for every $v \in C$
    $H(v) \leftarrow w \cdot f(v)$ // compute heuristic value of v
  Order C according to H and the total ordering $<_i$
  $B \leftarrow$ the first b nodes in C
  if $B \cap \bar{P} = \emptyset$ **then**
    $w \leftarrow w + \alpha \cdot \left(\frac{\sum_{v^* \in \bar{P} \cap C} f(v^*)}{|\bar{P} \cap C|} - \frac{\sum_{v \in B} f(v)}{b} \right)$
    $B \leftarrow \bar{P} \cap C$
return

Figure 8: Online algorithm for best-first beam search.

updating weights given the current beam and recomputing the beam given the updated weights. Just as traditional EM is quite prone to local minima, so are the LaSO algorithms in general, and in particular even when there is a positive search margin as demonstrated in the following counter example. Note that the standard Perceptron algorithm for classification learning does not run into this problem since there is no hidden state involved.

Counter Example 1 *We give a training set for which the existence of a weight vector with positive search margin does not guarantee convergence to a solution weight vector for LaSO-BR or LaSO-BST. Consider a problem that consists of a single training instance with search space shown in Figure 9, preference ordering $C < B < F < E < D < H < G$, and single target path $P = (\{A\}, \{B\}, \{E\})$.*

First we will consider using breadth-first beam search with a beam width of $b = 2$. Using the weight vector $w = [\gamma, \gamma]$ the resulting beam trajectory will be (note that higher values of $w \cdot f(v)$ are better):

$$\{A\}, \{B,C\}, \{E,F\}.$$

The search margin of w, which is only sensitive to pairs of target and non-target nodes that were compared during the search, is equal to,

$$\gamma = w \cdot f(B) - w \cdot f(C) = w \cdot f(E) - w \cdot f(F)$$

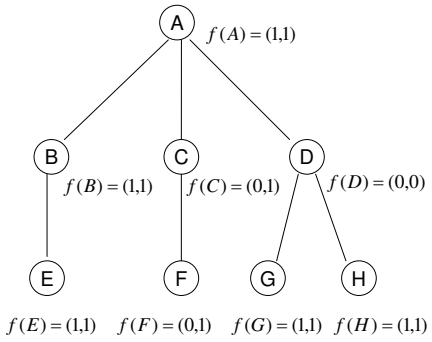


Figure 9: Counter example for convergence under positive search margin.

which is positive. We now show that the existence of w does not imply convergence under perceptron updates.

Consider simulating LaSO-BR starting from $w' = 0$. *The first search step gives the beam* $\{D,B\}$ *according to the given preference ordering. Since B is on the target path we continue expanding to the next level where we get the new beam* $\{G,H\}$. *None of the nodes are on the target path so we update the weights as follows:*

$$\begin{aligned} w' &= w' + f(E) - 0.5[f(G) + f(H)] \\ &= w' + [1,1] - [1,1] \\ &= w'. \end{aligned}$$

This shows that w' *does not change and we have converged to the weight vector* $w' = 0$, *which is not a solution to the problem.*

For the case of best-first beam search, the performance is similar. Given the weight vector $w = [\gamma, \gamma]$, *the resulting beam search with beam width 2 will generate the beam sequence,*

$$\{A\}, \{B,C\}, \{E,C\}$$

which is consistent with the target trajectory. From this we can see that w has a positive search margin of:

$$\gamma = w \cdot f(B) - w \cdot f(C) = w \cdot f(E) - w \cdot f(C).$$

However, if we follow the perceptron algorithm when started with the weight vector $w' = 0$ *we can again show that the algorithm does not converge to a solution weight vector. In particular, the first search step gives the beam* $\{D,B\}$ *and since B is on the target path, we do not update the weights and generate a new beam* $\{G,H\}$ *by expanding the node D. At this point there are no target nodes in the beam and the weights are updated as follows*

$$\begin{aligned} w' &= w' + f(B) - 0.5[f(G) + f(H)] \\ &= w' + [1,1] - [1,1] \\ &= w' \end{aligned}$$

showing that the algorithm has converged to $w' = 0$, which is not a solution to the problem.

Thus, we have shown that a positive search margin does not guarantee convergence for either LaSO-BR or LaSO-BST. This counter example also applies to the original LaSO algorithm, which is quite similar to LaSO-BST.

4.3 Convergence Under Stronger Notions of Margin

Given that linear separability, or equivalently a positive search margin, is not sufficient to guarantee convergence we consider a stronger notion of margin, the *level margin*, which measures by how much the target nodes are ranked above (or below) other non-target nodes at the same search level.

Definition 11 (Level Margin) *The level margin of a weight vector w for a training set is defined as $\gamma = min_{\{(v^*,v)\}}(w \cdot f(v^*) - w \cdot f(v))$, where the set $\{(v^*, v)\}$ contains any pair such that v^* is a target node at some depth j and v can be reached in j search steps from the initial search node—that is, v^* and v are at the same level.*

For breadth-first beam search, a positive level margin for w implies a positive search margin, but not necessarily vice versa, showing that level margin is a strictly stronger notion of separability. The following result shows that a positive level margin is sufficient to guarantee convergence of LaSO-BR. Throughout we will let R be a constant such that for all training instances, for all nodes v and v', $\|f(v) - f(v')\| \leq R$. The proof of this result follows similar lines as the Perceptron convergence proof for standard classification problems Rosenblatt (1962).

Theorem 12 *Given a dead-end free training set such that there exists a weight vector w with level margin $\gamma > 0$ and $\|w\| = 1$, LaSO-BR will converge with a consistent weight vector after making no more than $(R/\gamma)^2$ weight updates.*

Proof First note that the dead-end free property of the training data can be used to show that unless the current weight vector is a solution it will eventually trigger a "meaningful" weight update (one where the candidate set contains target nodes).

Let w^k be the weights before the k'th mistake is made. Then $w^1 = 0$. Suppose the k'th mistake is made for the training data $\langle S_i, P_i \rangle$, when $B \cap P_{i,j} = \emptyset$. Here, $P_{i,j}$ is the j'th element of P_i, B is the beam generated at depth j for S_i and C is the candidate set from which B is selected. Note that C is generated by expanding all nodes in the previous beam and at least one of them is in $P_{i,j-1}$. With the dead-end free property, we are guaranteed that $C' = P_{i,j} \cap C \neq \emptyset$. The occurrence of the mistake indicates that, $\forall v^* \in P_{i,j} \cap C, v \in B$, $w^k \cdot f(v^*) \leq w^k \cdot f(v)$, which lets us derive an upper bound for $\|w^{k+1}\|^2$.

$$
\begin{aligned}
\|w^{k+1}\|^2 &= \|w^k + \frac{\sum_{v^* \in C'} f(v^*)}{|C'|} - \frac{\sum_{v \in B} f(v)}{b}\|^2 \\
&= \|w^k\|^2 + \|\frac{\sum_{v^* \in C'} f(v^*)}{|C'|} - \frac{\sum_{v \in B} f(v)}{b}\|^2 \\
&\quad + 2w^k \cdot (\frac{\sum_{v^* \in C'} f(v^*)}{|C'|} - \frac{\sum_{v \in B} f(v)}{b}) \\
&\leq \|w^k\|^2 + \|\frac{\sum_{v^* \in C'} f(v^*)}{|C'|} - \frac{\sum_{v \in B} f(v)}{b}\|^2 \\
&\leq \|w^k\|^2 + R^2
\end{aligned}
$$

where the first equality follows from the definition of the perceptron-update rule, the first inequality follows because $w^k \cdot (f(v^*) - f(v)) < 0$ for all $v^* \in C', v \in B$, and the second inequality follows from the definition of R. Using this upper-bound we get by induction that

$$\|w^{k+1}\|^2 \leq kR^2.$$

Suppose there is a weight vector w such that $||w|| = 1$ and w has a positive level margin, then we can derive a lower bound for $w \cdot w^{k+1}$.

$$\begin{aligned} w \cdot w^{k+1} &= w \cdot w^k + w \cdot (\frac{\sum_{v^* \in C'} f(v^*)}{|C'|} - \frac{\sum_{v \in B} f(v)}{b}) \\ &= w \cdot w^k + \frac{\sum_{v^* \in C'} w \cdot f(v^*)}{|C'|} - \frac{\sum_{v \in B} w \cdot f(v)}{b} \\ &\geq w \cdot w^k + \gamma. \end{aligned}$$

This inequality follows from the definition of the level margin γ of the weight vector w.

By induction, we get that $w \cdot w^{k+1} \geq k\gamma$. Combining this result with the above upper bound on $\|w^{k+1}\|$ and the fact that $\|w\| = 1$ we get that

$$1 \geq \frac{w \cdot w^{k+1}}{\|w\|\|w^{k+1}\|} \geq \sqrt{k}\frac{\gamma}{R} \Rightarrow k \leq \frac{R^2}{\gamma^2}.$$

Without the dead-end free property, LaSO-BR might generate a candidate set that contains no target nodes, which would allow for a mistake that does not result in a weight update. However, for a dead-end free training set, it is guaranteed that the weights will be updated if and only if a mistake is made. Thus, the mistake bound is equal to the bound on the weight updates. ■

Note that for the example search space in Figure 9 there is no weight vector with a positive level margin since the final layer contains target and non-target nodes with identical weight vectors. Thus, the non-convergence of LaSO-BR on that example is consistent with the above result. Unlike LaSO-BR, LaSO-BST and LaSO do not have such a guarantee since their beams can contain nodes from multiple levels. This is demonstrated by the following counter example.

Counter Example 2 *We give a training set for which the existence of a w with positive level margin does not guarantee convergence for LaSO-BST. Consider a single training example with the search space in Figure 10, single target path $P = (\{A\}, \{B\}, \{E\})$, and preference ordering $C < B < E < F < G < D$.*

Given the weight vector $w = [2\gamma, \gamma]$, the level margin of w is equal to γ. However, starting with $w' = 0$ and running LaSO-BST the first search step gives the beam $\{D, B\}$. Since B is on the target path, we get the new beam $\{G, F\}$ by expanding the node D. This beam does not contain a target node, which triggers the following weight update:

$$\begin{aligned} w' &= w' + f(B) - [f(F) + f(G)]/2 \\ &= w' + [1,0] - [1,0] \\ &= w'. \end{aligned}$$

Since w' does not change the algorithm has converged to $w' = 0$, which is not a solution to this problem. This shows that a positive level margin is not sufficient to guarantee the convergence of LaSO-BST. The same can be shown for the original LaSO.

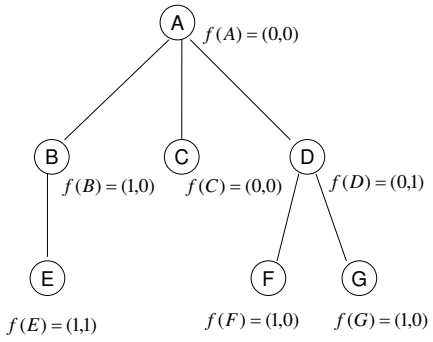


Figure 10: Counter example to convergence under positive level margin.

In order to guarantee convergence of LaSO-BST, we require an even stronger notion of margin, *global margin*, which measures the rank difference between any target node and any non-target node, regardless of search space level.

Definition 13 (Global Margin) *The global margin of a weight vector w for a training set is defined as* $\gamma = min_{\{(v^*,v)\}}(w \cdot f(v^*) - w \cdot f(v))$*, where the set* $\{(v^*,v)\}$ *contains any pair such that* v^* *is any target node and v is any non-target node in the search space.*

Note that if w has a positive global margin then it has a positive level margin. The converse is not necessarily true. The global margin is similar to the common definitions of margin used to characterize the convergence of linear perceptron classifiers (Novikoff, 1962).

To ensure convergence of LaSO-BST we also assume that the search spaces are all finite trees. This avoids the possibility of infinite best-first beam trajectories that never terminate at a goal node. Tree structures are quite common in practice and it is often easy to transform a finite search space into a tree. The structured classification experiments of Daumé III and Marcu (2005) and our own automated experiments both involve tree structured spaces.

Theorem 14 *Given a dead-end free training set of finite tree search spaces such that there exists a weight vector w with global margin* $\gamma > 0$ *and* $\|w\| = 1$*, LaSO-BST will converge with a consistent weight vector after making no more than* $(R/\gamma)^2$ *weight updates.*

The proof is similar to that of Theorem 12 except that the derivation of the lower bound makes use of the global margin and we must verify that the restriction to finite tree search spaces guarantees that each iteration of LaSO-BST will terminate with a goal node being reached. We were unable to show convergence for the original LaSO algorithm even under the assumptions of this theorem.

In summary, this section has introduced three different notions of margin: search margin, level margin, and global margin. Both algorithms converge for a positive global margin, which implies a positive search margin and a positive level margin. For LaSO-BR, but not LaSO-BST, convergence is guaranteed for a positive level margin, which implies a positive search margin. This shows that LaSO-BR converges under a strictly weaker notion of margin than LaSO-BST due to the fact that the ranking decisions of breadth-first search are restricted to nodes at the same level of the search space, as opposed to best-first search. This suggests that it may often be easier to define effective feature spaces for the breadth-first paradigm. Finally, a positive search margin corresponds exactly

to linear separability, but is not enough to guarantee convergence for either algorithm. This is in contrast to results for linear classifier learning, where linear separability implies convergence of perceptron updates.

4.4 Convergence for Ambiguous Training Data

Here we study convergence for linearly inseparable training data. Inseparability is often the result of training-data ambiguity, in the sense that many "good" solution paths are not included as target paths. For example, this is common in automated planning where there can be many (nearly) optimal solutions, many of which are inherently identical (e.g., differing in the orderings of unrelated actions). It is usually impractical to include all solutions in the training data, which can make it infeasible to learn a ranking function that strictly prefers the target paths over the inherently identical paths not included as targets. In these situations, the above notions of margin will all be negative. Here we consider the notion of *beam margin* that allows for some amount of ambiguity, or inseparability.

For each instance $\langle S_i, P_i \rangle$, where $S_i = \langle I_i, s_i(\cdot), f(\cdot), <_i \rangle$ and $P_i = (P_{i,1}, P_{i,2}, \ldots, P_{i,d_i})$, let D_{ij} be the set of nodes that can be reached in j search steps from I_i. That is, D_{ij} is the set of all possible non-target nodes that could be in beam $B_{i,j}$. A beam margin is a triple (b', δ_1, δ_2) where b' is a non-negative integer, and $\delta_1, \delta_2 \geq 0$.

Definition 15 (Beam Margin) *A weight vector w has beam margin (b', δ_1, δ_2) on a training set $\{\langle S_i, P_i \rangle\}$, if for each i, j there is a set $D'_{ij} \subseteq D_{ij}$ such that $|D'_{ij}| \leq b'$ and*

$$\forall v^* \in P_{i,j}, v \in D_{ij} - D'_{ij}, \;\; w \cdot f(v^*) - w \cdot f(v) \geq \delta_1 \text{ and,}$$
$$\forall v^* \in P_{i,j}, v \in D'_{ij}, \;\; \delta_1 > w \cdot f(v^*) - w \cdot f(v) \geq -\delta_2.$$

A weight vector w has beam margin (b', δ_1, δ_2) if at each search depth it ranks the target nodes better than most other non-target nodes (those in $D_{ij} - D'_{ij}$) by a margin of at least δ_1, and ranks at most b' non-target nodes (those in D'_{ij}) better than the target nodes by a margin no greater than δ_2. Whenever this condition is satisfied we are guaranteed that a beam search of width $b > b'$ guided by w will solve all of the training problems. The case where $b' = 0$ corresponds to the level margin, where the data is separable. By allowing $b' > 0$ we can consider cases where there is no "dominating" weight vector that ranks all targets better than all non-targets at the same level. The following result shows that for a large enough beam width, which is dependent on the beam margin, LaSO-BR will converge to a consistent solution.

Theorem 16 *Given a dead-end free training set, if there exists a weight vector w with beam margin (b', δ_1, δ_2) and $\|w\| = 1$, then for any beam width $b > (1 + \delta_2/\delta_1) b' = b^*$, LaSO-BR will converge with a consistent weight vector after making no more than $(R/\delta_1)^2 \left(1 - b^* b^{-1}\right)^{-2}$ weight updates.*

Proof Let w^k be the weights before the k'th mistake is made, so that $w^1 = 0$. Suppose that the k'th mistake is made when $B \cap P_{i,j} = \emptyset$ where B is the beam generated at depth j for the ith training instance. We can derive the upper bound of $\|w^{k+1}\|^2 \leq kR^2$ as in the proof of Theorem 12.

Next we derive a lower bound on $w \cdot w^{k+1}$. Denote by $B' \subseteq B$ the set of nodes in the beam such that $\delta_1 > w \cdot (f(v^*) - f(v)) \geq -\delta_2$ and let $C' = P_{i,j} \cap C$. By the definition of beam margin, we have $|B'| < b'$.

$$
\begin{aligned}
w \cdot w^{k+1} &= w \cdot w^k + w \cdot \left(\frac{\sum_{v* \in C'} f(v^*)}{|C'|} - \frac{\sum_{v \in B} f(v)}{b}\right) \\
&= w \cdot w^k + w \cdot \sum_{v \in B - B'} \frac{\frac{\sum_{v* \in C'} f(v^*)}{|C'|} - f(v)}{b} \\
&\quad + w \cdot \sum_{v \in B'} \frac{\frac{\sum_{v* \in C'} f(v^*)}{|C'|} - f(v)}{b} \\
&\geq w \cdot w^k + \frac{(b-b')\delta_1}{b} - \frac{b'\delta_2}{b}.
\end{aligned}
$$

By induction, we get that $w \cdot w^{k+1} \geq k\frac{(b-b')\delta_1 - b'\delta_2}{b}$. Combining this result with the above upper bound on $\|w^{k+1}\|$ and the fact that $\|w\| = 1$ we get that $1 \geq \frac{w \cdot w^{k+1}}{\|w\|\|w^{k+1}\|} \geq \sqrt{k}\frac{(b-b')\delta_1 - b'\delta_2}{bR}$. The mistake bound follows by noting that $b > b^*$ and algebra.

Similar to Theorem 12, the dead-end free property of the training set guarantees that the mistake bound is equal to the bound on the weight updates. ■

Note that when there is a positive level margin (i.e., $b' = 0$), the mistake bound here reduces to $(R/\delta_1)^2$, which does not depend on the beam width and matches the result for separable data. This is also the behavior when $b >> b^*$.

An interesting aspect of this result is that the mistake bound depends on the beam width. Rather, all of our previous convergence results were independent of the beam width and held even for beam width $b = 1$. Thus, those previous results did not provide any formalization of the intuition that the learning problem will often become easier as the beam width increases, or equivalently as the amount of search increases. Indeed, in the extreme case of exhaustive search, no learning is needed at all, whereas for $b = 1$ the ranking function has little room for error.

To get a sense for the dependence on the beam width consider two extreme cases. As noted above, for very large beam widths such that $b >> b^*$, the bound becomes $(R/\delta_1)^2$. On the other extreme, if we assume $\delta_1 = \delta_2$ and we use the smallest possible beam width allowed by the theorem $b = 2b' + 1$, then the bound becomes $((2b'+1)R/\delta_1)^2$, which is a factor of $(2b'+1)^2$ larger than when $b >> b'$. This shows that as we increase b (i.e., the amount of search), the mistake bound decreases, suggesting that learning becomes easier, agreeing with intuition.

It is also possible to define an analog to the beam margin for best first beam search. However, in order to guarantee convergence, the conditions on ambiguity would be relative to the global state space, rather than local to each level of the search space.

5. Application to Automated Planning

In this section, we present an empirical evaluation of beam-search learning in the context of automated planning. We first give related background, followed by the technical details regarding our application to automated planning. Then, we present the experimental results.

5.1 Background

Here we give background related to automated planning, the problem of learning to plan, and prior related work in the area of learning to plan.

5.1.1 AUTOMATED PLANNING

Planning is a subfield of artificial intelligence that studies algorithms for selecting sequences of actions in order to achieve goals. In this work, we consider planning domains and planning problems described using the STRIPS fragment of the planning domain description language (PDDL) (McDermott, 1998), which we now outline.

A planning domain $\mathcal{D}$ defines a set of possible actions $\mathcal{A}$ and a set of world states $\mathcal{W}$ in terms of a set of predicate symbols P, action types Y, and constants C. A state fact is the application of a predicate to the appropriate number of constants, with a state being a set of state facts. Each action $a \in \mathcal{A}$ consists of: 1) an action name, which is an action type applied to the appropriate number of constants, 2) a set of precondition state facts $\text{Pre}(a)$, 3) two sets of state facts $\text{Add}(a)$ and $\text{Del}(a)$ representing the add and delete effects respectively. An action a is applicable to a world state ω iff $\text{Pre}(a) \subseteq \omega$, and the application of an (applicable) action a to ω results in the new state $\omega' = (\omega \setminus \text{Del}(a)) \cup \text{Add}(a)$. That is, the application of an action adds the facts in the add list to the state and deletes facts in the delete list.

Given a planning domain, a planning problem is a tuple (ω, A, g), where $A \subseteq \mathcal{A}$ is a set of actions, $\omega \in \mathcal{W}$ is the initial state, and g is a set of state facts representing the goal. A solution plan for a planning problem is a sequence of actions $\langle a_1, \ldots, a_l \rangle$, where the sequential application of the sequence starting in state ω leads to a goal state ω' where $g \subseteq \omega'$. In this paper, we will view planning problems as directed graphs where the vertices represent states and the edges represent possible state transitions. Planning then reduces to graph search for a path from the initial state to goal.

Figure 1 shows an example of the search space corresponding to a problem from the Blocksworld planning domain. Here, the initial state is described by the facts

$$\begin{aligned}\omega_0 = \{&clear(A), clear(B), clear(C), clear(D), ontable(A),\\ &ontable(B), ontable(C), ontable(D), armempty\}.\end{aligned}$$

An example action from the domain is $pickup(A)$ with the following definition:

$$\begin{aligned} Pre(pickup(A)) &= \{clear(A), ontable(A), armempty\}\\ Add(pickup(A)) &= \{holding(A)\}\\ Del(pickup(A)) &= \{clear(A), ontable(A), armempty\}.\end{aligned}$$

Note that the precondition of this action is satisfied in ω_0 and hence can be applied from ω_0, which would result in the new state

$$\omega_1 = \{holding(A), clear(B), clear(C), clear(D), ontable(B), ontable(C), ontable(D)\}.$$

If the goal of the planning problem is $g = \{on(C,D), on(B,A), clear(C), clear(B)\}$, then one solution for the problem, as shown in Figure 1, is the action sequence $\langle pickup(B), stack(B,A), pickup(C), stack(C,D)\rangle$.

There has been much recent progress in automated planning. One of the most successful approaches, and the one most relevant to this paper, is to solve planning problems using forward state-space search guided by powerful domain-independent planning heuristics. A number of recent state-of-the-art planners have followed this approach including HSP (Bonet and Geffner, 1999), FF (Hoffmann and Nebel, 2001), and AltAlt (Nguyen et al., 2002) to name just a few.

5.1.2 LEARNING TO PLAN

It is common for planning systems to be asked to solve many problems from a particular domain. For example, the bi-annual international planning competition is organized around a number of planning domains and includes many problems of varying difficulty from each domain. Given that problems from the same domain share significant structure, it is natural to attempt to learn from past experience in a domain in order to solve future problems from the same domain more efficiently. However, most state-of-the-art planning systems do not have any such learning capability and rather solve each problem from the domain as if it were the first problem ever encountered by the planner. The goal of our work is to develop the capability for a planner to learn domain-specific knowledge in order to improve performance in a target domain of interest.

More specifically, we focus on developing learning capabilities within the simple, but highly successful, framework of heuristic state-space search planning. Our goal is to learn heuristics, or ranking functions, that can quickly solve problems using beam search with a small beam width. Given a representative training set of problems from a planning domain, our approach first solves the problems using potentially expensive search (e.g., using a large beam width), guided by an existing heuristic. These solutions are then used to learn a heuristic that can guide a small width beam search to the same solutions. The hope is that the learned heuristic will then generalize and allow for the quick solution of new problems that could not be practically solved before learning.

5.1.3 PRIOR WORK

There has been a long history of work on learning-to-plan, originating at least back to the original STRIPS planner (Fikes et al., 1972), which learned triangle tables or macros that could later be exploited by the planner. For a collection and survey of work on learning in AI planning see Minton (1993) and Zimmerman and Kambhampati (2003).

A number of learning-to-plan systems have been based on the explanation-based learning (EBL) paradigm, for example, Minton et al. (1989) among many others. EBL is a deductive learning approach, in the sense that the learned knowledge is provably correct. Despite the relatively large effort invested in EBL research, the best approaches typically did not consistently lead to significant gains, and even hurt performance in many cases. A primary way that EBL can hurt performance is by learning too many, overly specific control rules, which results in the planner spending too much time simply evaluating the rules at the cost of reducing the number of search nodes considered. This problem is commonly referred to as the EBL utility problem (Minton, 1988).

Partly in response to the difficulties associated with EBL-based approaches, there have been a number of systems based on inductive learning, sometimes combined with EBL. The inductive approach involves applying statistical learning mechanisms in order to find common patterns that can distinguish between good and bad search decisions. Unlike EBL, the learned control knowledge typical does not have guarantees of correctness, however, the knowledge is typically more general and hence more effective in practice. Some representative examples of such systems include

learning for partial-order planning (Estlin and Mooney, 1996), learning for planning as satisfiability (Huang et al., 2000), and learning for the Prodigy means-ends framework (Aler et al., 2002). While these systems typically showed better scalability than their EBL counterparts, the evaluations were typically conducted on only a small number of planning domains and/or small number of test problems. There is no empirical evidence that such systems are robust enough to compete against state-of-the-art non-learning planners across a wide range of domains.

More recently there have been several learning-to-plan systems based on the idea of learning reactive policies for planning domains (Khardon, 1999; Martin and Geffner, 2000; Yoon et al., 2002). These approaches use statistical learning techniques to learn policies, or functions, that map any state-goal pair from a given domain to an appropriate action. Given a good reactive policy for a domain, problems can be solved quickly, without search, by iterative application of the policy. Despite its simplicity, this approach has demonstrated considerable success. However, these approaches have still not demonstrated the robustness necessary to outperform state-of-the-art non-learning planners across a wide range of domains.

More closely related is work by La Rosa et al. (2007), which uses a case-based reasoning approach to obtained an improved heuristic for forward state-space search. It is likely that our weight learning approach could be combined with their system to harness the benefits of both approaches. The most closely related approach to our work is based on extending forward state-space search planners by learning improved heuristics (Yoon et al., 2006), an approach which is among the state-of-the-art among learning-based planners. That work focused on improving the relaxed plan length heuristic used by the state-of-the-art planner FF (Hoffmann and Nebel, 2001). Note that FF consists of two stages: an incomplete local search and a complete best first search. In particular, Yoon et al. (2006) applied linear regression to learn an approximation of the difference between FF's heuristic and the observed distances-to-goal of states in the training plans. The primary contribution of the work was to define a generic knowledge representation for features and a features-search procedure that allowed learning of good regression functions across a range of planning domains. While the approach showed promising results, the learning mechanism has a number of potential shortcomings. Most importantly, the mechanism does not consider the actual search performance of the heuristic during learning. That is, learning is based purely on approximating the observed distances-to-goal in the training data. Even if the learned heuristic performs poorly on the training data when used for search, the learner makes no attempt to correct the heuristic in response.

A primary motivation for this paper is to develop a heuristic learning mechanism that is more tightly integrated with the search process. Our LaSO-style algorithms for learning beam-search ranking functions do exactly that. Our learning approach can be viewed as error-driven in the sense that it directly attempts to correct errors as they arise in the search process, rather than attempting to precisely model the distance-to-goal. In many areas of machine learning, such error-driven methods have been observed to outperform their traditional passive counterparts. The experimental results presented here agree with that observation in a number of planning domains.

5.2 Experimental Setup

We present experiments in eight STRIPS domains: Blocksworld, Pipesworld, Pipesworld-with-tankage, PSR, Philosopher, DriverLog, Depots and FreeCell. All of these domains with the exception of Blocksworld were taken from the 3rd and 4th international planning competitions (IPC3 and IPC4). With only two exceptions, this is the same set of domains used to evaluate the approach of

Yoon et al. (2006), which is the only prior work that we are aware of for learning heuristics to improve forward state-space search in automated planning. The difference between our set of domains and theirs is that we include Blocksworld, while they did not, and we do not include the Optical Telegraph domain, while they did. Our reason for not showing results for Optical Telegraph is that none of the systems we evaluated were able to solve any of the problems.[1]

5.2.1 DOMAIN PROBLEM SETS

For each domain we needed to create a set of training problems and testing problems on which the learned heuristics would be trained and evaluated. In Blocksworld, all problems were generated using the BWSTATES generator (Slaney and Thiébaux, 2001), which produces random Blocksworld problems. Thirty problems with 10 or 20 blocks were used as training data, and 30 problems with 20, 30, or 40 blocks were used for testing. For DriverLog, Depots and FreeCell, the first 20 problems are taken from IPC3 and we generated 30 more problems of varying difficulty to arrive at a set of 50 problems, roughly ordered by difficulty. For each domain, we used the first 15 problems for training and the remaining 35 for testing. The other four domains are all taken from IPC4. Each domain includes 50 or 48 problems, roughly ordered by difficulty. In each case, we used the first 15 problems for training and the remaining problems for testing.

5.2.2 SEARCH SPACE DEFINITION

We now describe the mapping between the planning problem described in Section 5.1.1 and the general search spaces described in Section 2, which were the basis for describing our algorithms. Recall that a general search space is a tuple $\langle I, s(\cdot), f(\cdot), < \rangle$ giving the initial state, successor function, feature function, and preference ordering respectively. In the context of planning each search node is a state-goal pair (ω, g), where ω can be thought of as the current world state, g is the current goal, and both are represented as sets of facts. Note that it is important that nodes contain both state and goal information, rather than just state information, since the evaluation/ranking of a search node depends on how good ω is with respect to the particular goal g. The initial search node I is equal to (ω_0, g), where ω_0 is the initial state of the planning problem and g is the problem's goal. The successor function s maps a search node (ω, g) to the set of all nodes of the form (ω', g) where ω' is a state that can be reached from ω via the application of some action whose preconditions are satisfied in ω. Note that according to this definition all nodes in a search space contain the same goal component. The feature function $f((\omega, g)) = (f_1((\omega, g)), \ldots, f_m((\omega, g)))$ can be any function over world states and goals. The particular functions we use in this work are describe later in this section. Finally, the preference ordering $<$ is simply the default ordering used by the planner FF, which is the planner our implementation is based on.

5.2.3 TRAINING DATA GENERATION

The LaSO-style algorithms learn from target solution paths, which requires that we generate solution plans for all of the training problems. To do this, for each training problem, we selected the shortest plan out of those found by running the planner FF and beam search with various large beam

1. The results in Yoon et al. (2006) indicated that their linear regression learning method was effective in Optical Telescope. Our implementation of linear regression, however, was unable to solve any of the problems. After investigating this difference, we found that it is due to a subtle difference in the way that ties are broken during forward state-space search, indicating that the linear regression method was not particularly robust in this domain.

widths guided by FF's relaxed-plan heuristic. The resulting plans are totally ordered sequences of actions and one could simply label each training problem by its corresponding sequence of actions. However, in many cases, it is possible to produce equivalent plans by permuting the order of certain actions in the totally ordered plans. That is, there are usually many other equivalent totally ordered plans. Thus, including only the single plan found via the above approach in the training data results in significant ambiguity in the sense described in Section 4.4.

In order to help reduce the ambiguity it is desirable to try to include as many equivalent plans as possible as part of the target plan set for a particular problem. To do this, instead of using just a single totally ordered solution plan in the training data for each problem, we transform each such totally ordered plan into a partial-order plan, which contains the same set of actions but only includes action-ordering constraints that appear to be necessary. Finding minimal partial-order plans from total-order plans is an NP-hard problem and hence we use the heuristic algorithm described in Veloso et al. (1991). For each training problem, the resulting partial-order plan provides an implicit representation for a potentially exponentially large set of solution trajectories. By using these partial-order plans as the labels for our training problems we can significantly reduce the ambiguity in the training data. In preliminary experiments, the performance of our learning algorithms improved in a number of domains when using training data that included the partial-order plans rather than the original total-order plans.

5.2.4 HEURISTIC REPRESENTATION AND DOMAIN FEATURES

We consider learning heuristic functions that are represented as weighted linear combinations of features, that is, $H(v) = \Sigma_i w_i \cdot f_i(v)$ where v is a search node, f_i is a feature of search nodes, and w_i is the weight of feature f_i. One of the challenges with this representation is to define a generic feature space from which features can be selected for each domain. This space must be rich enough to capture important properties of a wide range of planning domains, but also be amenable to searching for those properties. For this purpose we will draw on prior work Yoon et al. (2008) that defined such a feature space using a first-order language.

Each feature in the above space is defined by a taxonomic class expression, which represents a set of constants/objects in the planning domain. For example, a simple taxonomic class expression for the Blocksworld planning domain is *clear*, which represents the set of blocks that are currently clear, that is, the set of blocks x such that $clear(x) \in \omega$ where the current search node is $v = (\omega, g)$. The respective feature value represented by a class expression is equals to the cardinality of the class expression when evaluated at a search node. For example, if we let f_1 be the feature represented by the class expression *clear* then $f_1((\omega, g))$ is simply the number of clear blocks in ω. So in the example states from Section 5.1.1, $f_1(v_0) = f_1((\omega_0, g)) = 4$ and $f_1(v_1) = f_1((\omega_1, g)) = 3$. A more complex example for this problem is $clear \cap gclear$, which represents the set of blocks that are clear in both the current state and the goal, that is, the set containing any block x such that $clear(x) \in \omega$ and $clear(x) \in g$. If f_2 represents the feature corresponding to this expression then in the example states from 5.1.1 we get that $f_2(v_0) = 2$ and $f_2(v_1) = 2$.

Since our work in this paper is focused on weight learning, we refer to Yoon et al. (2008) for the full definition of the taxonomic feature language. Here we simply use a set of taxonomic features that have been automatically learned in prior work (Yoon et al., 2008) and tune their weights. In our experiments, this prior work gave us 15 features in Blocksworld, 35 features in Pipesworld, 11 features in Pipesworld-with-tankage, 54 features in PSR, 19 features in Philosopher, 22 features in

DriverLog, 3 features in Depot and 3 features in FreeCell. In all cases, we include FF's relaxed-plan-length heuristic as an additional feature.

5.3 Experimental Results

For each domain, we use LaSO-BR to learn weights with a learning rate of 0.01 for beam widths 1, 10, 50, and 100 and we will denote LaSO-BR run with beam width b by LaSO-BR$_b$. The maximum number of LaSO-BR iterations was set to 5000. In the evaluation procedure, we set a time cut-off of 30 CPU minutes per problem and considered a problem to be unsolved if a solution was not found within the cut-off.

In preliminary work, we also tried to apply LaSO-BST to our problems. However, this turned out to be an impractical approach due to the large potential search depths of these problems. In particular, we found that in many cases LaSO-BST would become stuck processing training examples, in the sense that it would neither update the weights nor make progress with respect to following the target trajectories. This typically occurred because LaSO-BST would maintain an early target node in the beam and thus not trigger a weight update, but at the same time would not progress to include deeper nodes on the target trajectories and instead explore paths off the target trajectories. To help remedy this behavior, we experimented with a variant of LaSO-BST that forces progress along the target trajectories after a specified number of search steps. For the Blocksworld planning domain and preliminary experiments in the other domains, we found that the results tended to improve compared to the original LaSO-BST, but still were not competitive with LaSO-BR. Thus for the experiments reported below we focus on LaSO-BR.

Note that the experiments in Daumé III and Marcu (2005) for structured classification produced good results using an algorithm very similar to LaSO-BST. There, however, the search spaces have small maximum depths (e.g., the length of a sentence), which apparently helped to avoid the problem we experienced here.

5.3.1 TRAINING TIME

Figure 11 gives the average training time required by LaSO-BR per iteration in each of our domains for four different beam widths. Note that Pipesworld was the only domain for which LaSO-BR converged to a consistent weight vector using a learning beam width 100. For all other training sets LaSO-BR never converged and thus terminated after 5000 iterations. The training time varies widely across the domains and depends on various factors including: number of features, number of actions, number of state predicates, and the number and length of target trajectories per training example. As expected the training times increase with the training beam width across the domains. It is difficult, however, to predict the relative times between different domains due to the complicated interactions among the above factors. Note that while these training times can be significant in many domains, the cost of training needs to only be paid once and then it is amortorized over all future problems. Furthermore, as we can observe later in the experimental results, a small beam width of 10 typically performs as well as larger widths.

5.3.2 DESCRIPTION OF TABLES

Before presenting our results we will first provide an overview of the information contained in our results tables. Figure 12 compares the performance of LaSO-BR$_{10}$ to three other algorithms,

Domain	$b = 1$	$b = 10$	$b = 50$	$b = 100$
Blocksworld	3	15	66	128
Pipesworld	1	4	13	24
Pipesworld-with-tankage	3	17	76	149
PSR	53	127	403	690
Philosopher	3	24	121	260
DriverLog	1	5	22	44
Depots	5	32	160	320
FreeCell	10	68	315	654

Figure 11: The average training time required by LaSO-BR per iteration for all training instances (seconds).

- LEN : beam search using FF's relaxed plan length heuristic
- U : beam search using a heuristic with uniform weights for all features
- LR : beam search using the heuristic learned from linear regression following the approach in Yoon et al. (2006).

We selected LaSO-BR$_{10}$ here because its performance is on par or better than other training beam widths. Note that in practice one could select the best beam width to use via cross-validation with a validation set of problems.

There is one table for each of our domains and each column in the tables is labeled by the algorithm used to generate the results. The rows correspond to the beam width used to generate the results on the testing problems, with the last row corresponding to using full best-first search (BFS) with an infinite beam width, which is the native search procedure used by FF. The columns are divided into three sets. The first four data columns labeled "Problems solved" give the number of problems solved using the testing beam width corresponding to the row, where a problem is considered solved if a solution is found within 30 minutes. The second set of columns labeled "Median plan length" gives the median length of solutions to the planning problems that were solved. The last 4 columns labeled "Median runtime " give the median runtime of each solver on the problems it solved. So, for example, the table shows that the heuristic learned via LaSO-BR$_{10}$ solves 26 Blocksworld test problems with a median solution length of 139 and a median runtime of 58.8 seconds using a testing beam width of 50, and solved 19 problems with a median solution length of 142 and a median runtime of 20.9 seconds using BFS.

Figure 13 is similar in structure to Figure 12 but compares the performance of heuristics learned using LaSO-BR with a variety of training beam widths and evaluated using a variety of testing beam widths. Only the number of problems solved and the median length of solutions that are found are considered here. For example, the upper left-most data point gives the number of problems solved using a learning beam width of 1 and a testing beam width of 1, while the first entry in the last column gives the median plan length of solved problems when learning with beam width 100 and testing with beam width 1.

Blocksworld												
	Problems solved				Median plan length				Median runtime (seconds)			
b	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$
1	13	0	11	**24**	3318	-	938	499	12.3	-	3.4	4.5
10	22	0	19	**24**	449	-	120	293	15.9	-	9.9	25.1
50	20	0	19	**26**	228	-	64	139	37.5	-	10.4	58.8
100	19	0	20	**24**	110	-	67	144	52.0	-	42.8	110.3
500	17	0	**23**	17	80	-	74	96	74.2	-	379.1	133.2
BFS	5	0	13	**19**	80	-	76	142	3.7	-	18.0	20.9

Pipesworld												
	Problems solved				Median plan length				Median runtime (seconds)			
b	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$
1	11	13	8	**16**	114	651	2476	2853	0.6	3.2	21.7	17.8
10	17	17	21	**23**	112	360	194	222	15.6	13.2	10.2	15.8
50	18	19	21	**26**	34	167	89	80	9.4	42.8	25.5	27.8
100	18	16	21	**24**	32	39	60	62	19.7	12.0	23.3	39.3
500	21	18	21	**25**	30	33	31	53	62.9	58.3	101.8	95.1
BFS	**15**	7	7	**15**	44	54	42	54	35.5	1.1	3.1	1.3

Pipesworld-with-tankage												
	Problems solved				Median plan length				Median runtime (seconds)			
b	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$
1	6	4	2	**7**	119	416	1678	291	8.0	18.2	92.1	8.4
10	6	8	**9**	8	68	603	399	117	70.2	256.5	125.6	33.4
50	6	5	6	**11**	61	111	94	122	358.4	281.4	186.1	116.3
100	5	4	5	**8**	54	105	43	55	482.4	279.4	255.5	190.6
500	5	6	4	**10**	42	97	41	76	938.5	586.1	210.7	492.0
BFS	**5**	3	2	3	59	60	126	100	431.2	17.1	935.7	22.0

PSR												
	Problems solved				Median plan length				Median runtime (seconds)			
b	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$
1	0	0	0	0	-	-	-	-	-	-	-	-
10	1	**20**	13	13	516	157	151	193	840.1	367.9	186.6	492.4
50	13	**17**	16	10	99	109	99	97	685.3	658.2	890.4	802.4
100	13	**15**	13	6	103	89	89	85	999.4	1121.9	1215.0	643.1
500	**4**	**4**	2	1	55	59	48	39	1035.6	1157.6	689.1	423.9
BFS	13	0	**21**	**21**	89	-	131	141	686.7	-	290.8	526.0

Philosopher												
	Problems solved				Median plan length				Median runtime (seconds)			
b	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$
1	0	**33**	**33**	**33**	-	363	363	363	-	12.5	18.1	13.3
10	0	**33**	**33**	11	-	363	363	1154	-	121.3	171.0	101.3
50	0	6	**23**	13	-	215	308	1579	-	77.6	387.4	825.1
100	0	16	**18**	6	-	292	281	1076	-	489.0	507.6	911.1
500	0	**7**	**7**	2	-	220	220	745	-	792.3	844.6	1280.7
BFS	0	**33**	**33**	0	-	363	363	-	-	9.5	329.8	-

DriverLog												
	Problems solved				Median plan length				Median runtime (seconds)			
b	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$
1	0	0	0	**8**	-	-	-	6801	-	-	-	364.2
10	3	0	0	**12**	789	-	-	1439	967.8	-	-	781.3
50	4	8	0	**12**	108	177	-	541	1199.3	457.6	-	998.5
100	1	**11**	0	**11**	98	147	-	275	1398.9	737.9	-	1131.6
500	0	**3**	0	1	-	86	-	94	-	1780.2	-	1237.1
BFS	**6**	2	0	1	162	181	-	138	1249.7	555.5	-	125.4

Depots												
	Problems solved				Median plan length				Median runtime (seconds)			
b	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$
1	1	1	2	**3**	462	790	411	790	3.9	6.5	3.8	6.7
10	4	1	4	**6**	195	28	981	3295	38.7	2.4	93.1	594.8
50	3	4	5	**6**	25	511	51	467	22.4	912.8	17.3	156.0
100	4	**7**	3	**7**	232	157	26	207	554.9	669.4	45.6	189.9
500	5	4	6	**11**	38	62	39	53	274.2	351.2	422.7	477.8
BFS	2	2	**3**	2	46	48	33	48	292.4	809.3	14.2	386.8

FreeCell												
	Problems solved				Median plan length				Median runtime (seconds)			
b	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$	LEN	U	LR	LaSO-BR$_{10}$
1	5	7	4	**9**	96	120	146	123	12.2	21.5	13.8	14.0
10	20	**22**	19	21	82	117	243	89	99.7	165.2	305.2	91.9
50	23	**24**	12	19	65	73	102	66	456.2	503.4	619.0	367.9
100	20	18	7	**21**	65	63	70	65	723.5	720.9	673.4	796.1
500	3	3	2	**4**	53	55	59	55	1400.0	1418.8	1518.5	1431.8
BFS	**23**	20	12	20	78	87	111	97	102.1	77.9	238.4	92.3

Figure 12: Experimental results for different planners. For each domain, we show the number of solved problems, the median plan length and median runtime of the solved problems. A dash in the table indicates that the planner was unable to solve any of the problems.

5.3.3 PERFORMANCE ACROSS TESTING BEAM WIDTHS

From Figure 12, in general, for all algorithms (learning and non-learning) we see that as the testing beam width begins to increase the number of solved problems and runtime increase and solution lengths improve. However, at some point as the beam width continues to increase the number of solved problems typically decreases. This behavior is typical for beam search, since as the testing beam width increases there is a greater chance of not pruning a solution trajectory, but the computational time and memory demands increase. Thus, for a fixed time cut-off we expect a decrease in performance as the beam width becomes large.

The median runtime typically increases as the test beam width increases, because more search nodes need to be evaluated. However, it is not always the case. The number of search nodes that are going to be evaluated also depends on the plan length. For example, while using LEN in the Depots planning domain, the median runtime of beam width 50 is smaller than that of beam width 10, because the median plan length improves from 195 to 25. Also note that it is not necessarily true that the plan lengths are strictly non-increasing with testing beam width. With large testing beam widths the number of candidates for the next beam increases, making it more likely for the heuristic to get confused by "bad" states. This is also one possible reason why performance tends to decrease with larger testing beam widths.

5.3.4 LASO-BR$_{10}$ VERSUS NO LEARNING

From Figure 12, we see that compared to LEN, the heuristic learned by LaSO-BR$_{10}$ tends to significantly improve the performance of beam search, especially for small beam widths. For example, in Blocksworld with beam width 1, LaSO-BR$_{10}$ solves almost twice as many problems as LEN. The median plan length has also been reduced significantly for beam width 1. As the beam width increases the gap between LaSO-BR$_{10}$ and LEN decreases but LaSO-BR$_{10}$ still solves more problems with comparable solution quality. In Pipesworld, LaSO-BR$_{10}$ has the best performance gap with beam width 50, solving 8 more problems than LEN. As the beam width increases, again the performance gap decreases, but LaSO-BR$_{10}$ consistently solves more problems than LEN. In this domain, the median plan lengths of LEN tend to be better, though a direct comparison of these lengths is not exactly fair since LaSO-BR$_{10}$ solves more problems, which are often the harder problems that result in longer plans. The trends with respect to number of solved problems are similar in other domains, with the exception of PSR and FreeCell. In PSR, LEN solves slightly more problems than LaSO-BR$_{10}$ at large beam widths. In FreeCell, LaSO-BR$_{10}$ is better than LEN for most case except for beam width 50.

These results show that LaSO-BR$_{10}$ is able to learn heuristics that significantly improve on the state-of-the-art heuristic LEN when using beam search. In general, the best performance was achieved for small beam widths close to those used for training, which is beneficial in terms of time and memory efficiency. Note that in practice one could use a validation set of problems in order to select the best combination of training beam width and testing beam width for a given domain. This is particularly natural in our current setting where our goal is to perform well relative to problems drawn from a given problem generator, in which case we can easily draw both training and evaluation problem sets.

5.3.5 COMPARING LASO-BR$_{10}$ WITH LINEAR REGRESSION

To compare with prior passive heuristic learning work we learned weights using linear regression following the approach of Yoon et al. (2006). To our knowledge this is the only previous system that addresses the heuristic learning problem in the context of forward state-space search in automated planning. In these experiments we used the linear regression tool available under Weka. The results for the resulting learned linear-regression heuristics are shown in the columns labeled LR in Figure 12.

For Blocksworld, LR solves fewer problems than LaSO-BR$_{10}$ with beam widths smaller than 500 but solves more problems than LaSO-BR$_{10}$ with beam width 500. The median plan length tends to favor LR except for the smallest beam width $b = 1$. For Pipesworld, DriverLog and Depots, LaSO-BR$_{10}$ always solves more problems than LR, with plan length again favoring LR to varying degrees. In Pipesworld-with-tankage, LaSO-BR$_{10}$ is better than LR for most case except for beam width 10, solving one less problem. In PSR and Philosopher, LR outperforms LaSO-BR$_{10}$ but LaSO-BR$_{10}$ achieves a comparable performance with small beam widths. In FreeCell, LaSO-BR$_{10}$ always solves more problems than LR with improved plan length.

These results indicate that error-driven learning can significantly improve over prior passive learning (here regression) in a number of domains. Indeed, there appears to be utility in integrating the learning process directly in the search procedure. However, the results also indicate that in some cases our current error-driven training method can fail to converge to a good solution in cases where regression happens to work well.

5.3.6 EFFECTS OF LEARNING BEAM WIDTH

Figure 13 compares the performance of LaSO-BR with different learning beam widths. For most domains, the performance doesn't change much as the learning beam width changes. Even with learning beam width 1, LaSO-BR can often achieve performance on par with larger learning beam widths. For example, in Blocksworld, LaSO-BR$_1$ results in the best performance at most testing beam widths except for beam width 500. For the other domains, LaSO-BR$_{10}$ typically is close to the performance of the best learning beam width. In a number of cases we see that LaSO-BR$_{10}$ performs significantly better than LaSO-BR$_{100}$, which suggests that learning with smaller beam widths can have some practical advantages. One reason for this might be due to the additional ambiguity in the weight updates when using larger beam widths. In particular, the weight update equations involve averages of all target and non-target nodes in the beams. The effect of this averaging is to effectively mix the feature vectors of large numbers of search nodes together. In many cases there will be a wide variety of non-target nodes in the beam, and this mixing can increase the difficulty of uncovering key patterns, which we conjecture might increase the requirements on training iterations and examples. In cases where the features are rich enough to support successful beam search with small width, it is then likely that learning with smaller widths will be better given a fixed number of iterations and examples. Note that the feature space we have used in this work has been previously demonstrated (Fern et al., 2006) to be particularly well suited to Blocksworld, which is perhaps one reason that $b = 1$ performed so well in that domain.

Finally note that contrary to what we originally expected it is not typically the case that the best performance for a particular testing beam width is achieved when learning with that same beam width. Rather the relationship between learning and testing beam widths is quite variable. Note that for most domains LaSO-BR never converged to a consistent weight vector in our experiments,

indicating that either the features were not powerful enough for consistency or the learning beam widths and/or number of iterations needed to be increased. In such cases, there is no clear technical reason to expect the best testing beam width to match the learning beam width. Thus, in general, we suggest the use of validation sets to select the best pair of learning and testing beam widths for a particular domain. Note that the lack of relationship between learning and test beam width is in contrast to that observed in Daumé III and Marcu (2005) for structured classification, where there appeared to be a small advantage to training and testing using the same width.

5.3.7 BEST FIRST SEARCH RESULTS

While our heuristic was learned for the purpose of controlling beam search we conducted one more experiment in each domain where we used the heuristics to guide Best First Search (*BFS*). We include these results primarily because BFS was the search procedure used to evaluate LR in Yoon et al. (2006) and is the native search strategy used by FF.[2] These results are shown in the bottom row of each table in Figure 12 and 13.

In Blocksworld, Pipesworld, PSR, LaSO-BR$_{10}$ was as good or better than the other three algorithms. Especially in Blocksworld, LaSO-BR$_{10}$ solves 19 problems while LEN only solves 5 problems. In Philosopher, neither LEN nor LaSO-BR$_{10}$ solves any problem. LEN is the best in Pipesworld-with-tankage, DriverLog and FreeCell, and LR works best in Depots. But for Pipesworld-with-tankage, Depots and FreeCell, the performance of LaSO-BR$_{10}$ is very close to the best planner.

These results indicate that the advantage of error-driven learning over regression is not just restricted to beam search, but appears to extend to other search approaches. That is, by learning in the context of beam search it is possible to extract problem solving information that is useful in other contexts.

5.3.8 PLAN LENGTH

LaSO-BR can significantly improve success rate at small beam widths, which is one of our main goals. However, the plan lengths at small widths are quite suboptimal, which is typical of beam search. Ideally we would like to obtain these success rates without paying a price in plan length. We are currently investigating ways to improve LaSO-BR in this direction. However, we note that typically one of the primary difficulties of automated planning is to simply find a path to the goal. After finding such a path, if it is significantly sub-optimal, incomplete plan analysis or plan rewriting rules can be used to significantly prune the plan, for example, see Ambite et al. (2000). Thus, despite the long plan lengths, the improved success rate of LaSO-BR at small beam widths could provide a good starting point for a fast plan length optimization.

6. Summary and Future Work

This paper presented a detailed study of the problem of learning ranking functions for beam search with an application to automated planning. On the theoretical side we first studied the computational complexity of this learning problem, highlighting the main dimensions of complexity by identifying core tractable and intractable subclasses. Next, we studied the convergence of recent online learning algorithms for this problem. The results clarified convergence issues, correcting and extending

2. FF actually uses two search strategies. In the first state it uses an incomplete strategy called enforced hill climbing. If that initial search does not find a solution then a best-first search is conducted.

Blocksworld								
	Problems solved				Median plan length			
b	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$
1	**27**	24	18	13	840	499	92	314
10	**27**	24	20	19	206	293	96	150
50	**27**	26	23	24	180	139	72	82
100	**25**	24	23	23	236	144	72	86
500	23	17	19	**24**	122	96	62	77
BFS	**21**	19	18	17	116	142	73	124

Pipesworld								
	Problems solved				Median plan length			
b	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$
1	16	16	**21**	15	1803	2853	1403	6958
10	**25**	23	23	21	227	222	179	270
50	25	**26**	25	22	74	80	119	75
100	**27**	24	23	22	146	62	104	47
500	23	**25**	20	21	60	53	61	37
BFS	14	**15**	13	8	59	54	103	42

Pipesworld-with-tankage								
	Problems solved				Median plan length			
b	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$
1	5	**7**	2	**7**	55	291	197	300
10	8	8	8	**10**	103	117	68	77
50	9	**11**	8	9	48	122	37	42
100	8	8	**10**	**10**	53	55	122	55
500	9	**10**	5	**10**	30	76	39	96
BFS	**6**	3	4	**6**	48	100	70	63

PSR								
	Problems solved				Median plan length			
b	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$
1	0	0	0	0	-	-	-	-
10	12	13	3	**14**	182	193	550	205
50	6	10	16	**17**	75	97	126	129
100	3	6	10	**13**	82	85	113	86
500	2	1	**4**	**4**	61	39	58	64
BFS	19	21	3	**25**	164	141	170	142

Philosopher								
	Problems solved				Median plan length			
b	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$
1	6	**33**	**33**	0	589	363	363	-
10	**19**	11	1	1	319	1154	451	1618
50	**13**	**13**	2	2	297	1579	1023	855
100	**9**	6	5	1	253	1076	255	1250
500	**4**	2	2	0	226	745	253	-
BFS	0	0	0	0	-	-	-	-

DriverLog								
	Problems solved				Median plan length			
b	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$
1	0	**8**	0	3	-	6801	-	4329
10	5	**12**	2	7	1227	1439	1061	435
50	0	**12**	1	1	-	541	129	136
100	0	**11**	0	1	-	275	-	98
500	0	**1**	0	0	-	94	-	-
BFS	1	1	0	**2**	154	138	-	332

Depots								
	Problems solved				Median plan length			
b	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$
1	**4**	3	2	2	1526	790	588	588
10	5	6	**7**	6	3259	3295	2042	715
50	2	6	**7**	3	517	467	707	392
100	4	**7**	6	5	43	207	147	54
500	6	**11**	**11**	5	47	53	53	38
BFS	**4**	2	2	2	106	48	48	48

FreeCell								
	Problems solved				Median plan length			
b	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$	LaSO-BR$_1$	LaSO-BR$_{10}$	LaSO-BR$_{50}$	LaSO-BR$_{100}$
1	7	**9**	5	5	132	123	125	133
10	**23**	21	**23**	19	89	89	85	71
50	**25**	19	24	24	69	66	68	68
100	24	21	22	**28**	68	65	65	72
500	**19**	4	21	**19**	61	55	62	61
BFS	23	20	**27**	25	104	97	104	104

Figure 13: Experimental results for various learning beam widths. For each domain, we show the number of solved problems and the median plan length of the solved problems. A dash in the table indicates that the planner was unable to solve any of the problems.

previous results. This included an analysis of convergence given ambiguous training data, giving a result that highlights the trade-off between the amount of allowed search and the difficulty of the resulting learning problem. Our experiments in the domain of automated planning showed that the approach has benefits compared to existing learning and non-learning state-space search planners. These results complement the positive empirical results in structured classification (Daumé III and Marcu, 2005) showing the general utility of the method.

In future work, we plan to extend the algorithms described here to allow for feature induction and more robust parameter estimation. We are also interested in studying learning in the context of search for other search strategies such as best-first and k-best-first search. In our initial investigations, we have found that the LaSO-style approach for these strategies has great difficulty in automated planning due to the very large depths of the search spaces, which makes it difficult to "assign credit" to search errors. This suggests that a key aspect of future work is to understand general credit-assignment mechanisms in the context of error-driven learning for search. Another important direction is to consider the application of these methods to new problem domains, in particular we are interested in more complex planning domains that include concurrency, durative actions, and uncertainty. It will also be interesting to consider learning beam-search heuristics for other search-based formulations of planning such as partial-order planning where the search is conducted directly in the space of partial-order plans.

Acknowledgments

Some of the material in this paper was first published at ICML-2007 (Xu and Fern, 2007) and IJCAI-07 (Xu et al., 2007).

Appendix A. Relation to Structured Classification

This Appendix assumes that the reader is familiar with the material in Section 3. The learning framework introduced in Section 2.2 is motivated by automated planning, with the objective of finding a goal node. It is important to note that the learning objective does not place a constraint on the rank of a goal node in the final beam compared to non-goal nodes, but rather only requires that there exists some goal node in the final beam. This is a natural formulation for automated planning where when solving test problems it is easy to test each beam to determine whether a goal node has been uncovered and to return a solution trajectory if one has. Thus, the exact ordering of the goal node in the final beam is not important with respect to finding solutions to planning problems.

In contrast, as described in the example at the end of Section 2.2, the formulation of structured classification as a search problem appears to require that we do pay attention to the rank of the goal nodes in the final beam. In particular, the formulation of Daumé III and Marcu (2005) requires the goal node to not only be contained in the final beam, but to be ranked higher than any other terminal node in the beam.

Since our formulation of the beam-search learning problem does not constrain the ranking of goal nodes relative to other nodes, it is not immediately clear how our formulation relates to structured classification. It turns out that these two formulations are polynomially equivalent, meaning that there is a polynomial reduction from each problem to the other. Thus, it is possible to compile away the explicit requirement that goal nodes have the highest rank in the final beam.

Below we adapt the definitions of the learning problems in Section 2.2 for structured classification. First, we introduce the notion of terminal node, which can be thought of as a possible solution to be returned by a structured classification algorithm, for example, a full parse tree for a sentence. We will denote the set of all terminal nodes as $\mathcal{T}$ and will assume a polynomial time test for determining whether a node is in this set. Note that some terminal nodes correspond to target solutions and others do not. When using beam search for structured classification the search is halted whenever a terminal node becomes highest ranked in the beam and the path leading to that terminal node is returned as the solution. Thus, successful learning must ensure both that no non-target terminal node ever becomes ranked first in any beam and also that eventually a target terminal node does become ranked first. This motivation leads to the following definitions for the breadth-first and best-first structured classification problems. Below, given the context of a weight vector w, we will denote the highest ranked node relative to w in a beam B by $B^{(1)}$.

Definition 17 (Breadth-First Structured Classification) *Given the input $\langle\{\langle S_i, P_i\rangle\}, b\rangle$, where b is a positive integer and $P_i = (P_{i,0}, \ldots, P_{i,d})$, the breadth-first structured classification problem asks us to decide whether there is a weight vector w such that for each S_i, the corresponding beam trajectory $(B_{i,0}, \ldots, B_{i,d})$, produced using w with a beam width of b, satisfies $B_{i,j} \cap P_{i,j} \neq \emptyset$ for each j, $B^{(1)}_{i,d} \in P_{i,d}$, and $B^{(1)}_{i,j} \notin \mathcal{T}$ for $j < d$?*

Definition 18 (Best-First Structured Classification) *Given the input $\langle\{\langle S_i, P_i\rangle\}, b\rangle$, where b is a positive integer and $P_i = (P_{i,0}, \ldots, P_{i,d})$, the best-first structured classification problem asks us to decide whether there is a weight vector w that produces for each S_i a beam trajectory $(B_{i,0}, \ldots, B_{i,k})$ of beam width b, such that $k \leq h$, each $B_{i,j}$ for $j < k$ contains at least one node in $\bigcup_j P_{i,j}$, $B^{(1)}_{i,k} \in P_{i,d}$, and $B^{(1)}_{i,j} \notin \mathcal{T}$ for $j < k$?*

We prove that these problems are polynomially equivalent to breadth-first and best-first consistency by showing that they are NP-complete. Since Section 3 proves that the consistency problems are also NP-complete we immediately get equivalence.

Theorem 19 *Breadth-first structured classification is NP-complete.*

Proof We can prove that the problem is in NP, following the structure of the proof of Theorem 4. Each certificate corresponds to a set of beam trajectories and has a size that is polynomial in the problem size. The certificate can be checked in polynomial time to see if for each i, it satisfies the conditions defined in Definition 17. From Lemma 3 in Section 3 we can then use the algorithm *TestTrajectories* in Figure 4 to decide whether there is a weight vector that generates the certificate in polynomial time. To show hardness we reduce from breadth-first consistency for the class of problems where $b = 1, d = 1, c = 6, t = 3$, and $n \geq 1$, which from Figure 6 is NP-complete. Since for this class the search spaces have depth 1 and the beam width is 1 it is easy to see that for any problem in this class, a weight vector is a solution to the consistency problem if and only if it is a solution to the structured classification problem. This shows that breadth-first structured classification is NP-hard and thus NP-complete. ■

Using an almost identical proof we can prove the same result for best-first structured classification.

Theorem 20 *Best-first structured classification is NP-complete.*

References

Shivani Agarwal and Dan Roth. Learnability of bipartite ranking functions. In *Proceedings of the Conference on Learning Theory*, 2005.

Ricardo Aler, Daniel Borrajo, and Pedro Isasi. Using genetic programming to learn and improve control knowledge. *Artificial Intelligence*, 141(1-2):29–56, 2002.

José Luis Ambite, Craig A. Knoblock, and Steven Minton. Learning plan rewriting rules. In *Proceeding of Artificial Intelligence Planning Systems*, pages 3–12, 2000.

Blai Bonet and Hećtor Geffner. Planning as heuristic search: New results. In *Proceedings of the European Conference on Planning*, pages 360–372, 1999.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with the perceptron algorithm. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002.

Hal Daumé III and Daniel Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the International Conference on Machine Learning*, 2005.

Tara A. Estlin and Rymond J. Mooney. Multi-strategy learning of search control for partial-order planning. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 843–848, 1996.

Alan Fern, Sungwook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias: Solving relational markov decision processes. *Journal of Artificial Intelligence Research*, 25:85–118, 2006.

Richard E. Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence Journal*, 3(1–3):251–288, 1972.

Michael R. Garey and David S. Johnson, editors. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

Klaus-Uwe Hoffgen, Hans-Ulrich Simon, and Kevin S. Van Horn. Robust trainability of single neurons. *Journal of Computer and System Sciences*, 50(1):114–125, 1995.

Jorg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:263–302, 2001.

Yi-Cheng Huang, Bart Selman, and Henry Kautz. Learning declarative control rules for constraint-based planning. In *Proceedings of Seventeenth International Conference on Machine Learning*, pages 415–422, 2000.

Rong Jin and Zoubin Ghahramani. Learning with multiple labels. In *Proceedings of the Sixteenth Annual Conference on Neural Information Processing Systems*, 2002.

Leonid G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20(1):191–194, 1979.

Roni Khardon. Learning action strategies for planning domains. *Artificial Intelligence*, 113(1-2): 125–148, 1999.

Tomás La Rosa, Angel García Olaya, and Daniel Borrajo. Using cases utility for heuristic planning improvement. In *Proceedings of the Seventh International Conference on Case Based Reasoning*, 2007.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289, 2001.

Mario Martin and Hector Geffner. Learning generalized policies in planning domains using concept languages. In *Proceedings of Seventh International Conference on Principles of Knowledge Representation and Reasoning*, 2000.

Drew McDermott. PDDL- the planning domain definition language. In *The 1st International Planning Competition*, 1998.

Steven Minton. Quantitative results concerning the utility of explanation-based learning. In *Proceedings of National Conference on Artificial Intelligence*, 1988.

Steven Minton, editor. *Machine Learning Methods for Planning*. Morgan Kaufmann Publishers, 1993.

Steven Minton, Jaime G. Carbonell, Craig A. Knoblock, Daniel Kuokka, Oren Etzioni, and Yolanda Gil. Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40: 63–118, 1989.

Xuanlong Nguyen, Subbarao Kambhampati, and Romeo S. Nigenda. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. *Artificial Intelligence*, 135(1-2):73–123, 2002.

Albert B. Novikoff. On convergence proofs on perceptrons. In *Symposium on the Mathematical Theory of Automata*, pages 615–622, 1962.

Frank Rosenblatt. *Principles of Neurodynamics*. Spartan, New York, 1962.

John Slaney and Sylvie Thiébaux. Blocks world revisited. *Artificial Intelligence*, 125:119–153, 2001.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Neural Information Processing Systems Conference*, 2003.

Manuela M. Veloso, M. Alicia Pérez, and Jamie G. Carbonell. Nonlinear planning with parallel resource allocation. In *Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 207–212, 1991.

Yuehua Xu and Alan Fern. On learning linear ranking functions for beam search. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2007.

Yuehua Xu, Alan Fern, and Sungwook Yoon. Discriminative learning of beam-search heuristics for planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2007.

Sungwook Yoon, Alan Fern, and Robert Givan. Inductive policy selection for first-order MDPs. In *Proceedings of Eighteenth Conference in Uncertainty in Artificial Intelligence*, 2002.

Sungwook Yoon, Alan Fern, and Robert Givan. Learning heuristic functions from relaxed plans. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2006.

Sungwook Yoon, Alan Fern, and Robert Givan. Learning control knowledge for forward search planning. *Journal of Machine Learning Research*, 9:683–718, 2008.

Terry Zimmerman and Subbarao Kambhampati. Learning-assisted automated planning: Looking back, taking stock, going forward. *AI Magazine*, 24(2)(2):73–96, 2003.

Journal of Machine Learning Research 10 (2009) 1611-1631 Submitted 5/08; Revised 2/09; Published 7/09

Marginal Likelihood Integrals for Mixtures of Independence Models

Shaowei Lin SHAOWEI@MATH.BERKELEY.EDU
Bernd Sturmfels BERND@MATH.BERKELEY.EDU
Department of Mathematics
University of California
Berkeley, CA 94720, USA

Zhiqiang Xu XUZQ@LSEC.CC.AC.CN
LSEC, Academy of Mathematics and System Sciences
Chinese Academy of Sciences
Beijing, 100080, China

Editor: Tommi Jaakkola

Abstract

Inference in Bayesian statistics involves the evaluation of marginal likelihood integrals. We present algebraic algorithms for computing such integrals exactly for discrete data of small sample size. Our methods apply to both uniform priors and Dirichlet priors. The underlying statistical models are mixtures of independent distributions, or, in geometric language, secant varieties of Segre-Veronese varieties.

Keywords: marginal likelihood, exact integration, mixture of independence model, computational algebra

1. Introduction

Evaluation of marginal likelihood integrals is central to Bayesian statistics. It is generally assumed that these integrals cannot be evaluated exactly, except in trivial cases, and a wide range of numerical techniques (e.g., MCMC) have been developed to obtain asymptotics and numerical approximations (Chickering and Heckerman, 1997). The aim of this paper is to show that exact integration is more feasible than is surmised in the literature. We examine marginal likelihood integrals for a class of mixture models for discrete data. Bayesian inference for these models arises in many contexts, including machine learning and computational biology. Recent work in these fields has made a connection to singularities in algebraic geometry (Drton, 2009; Geiger and Rusakov, 2005; Watanabe, 2001; Watanabe and Yamazaki, 2003, 2004). Our study augments these developments by providing tools for symbolic integration when the sample size is small.

The numerical value of the integral we have in mind is a rational number, and exact evaluation means computing that rational number rather than a floating point approximation. For a first example consider the integral

$$\int_{\Theta} \prod_{i,j \in \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}} \left(\pi \lambda_i^{(1)} \lambda_j^{(2)} + \tau \rho_i^{(1)} \rho_j^{(2)}\right)^{U_{ij}} d\pi\, d\tau\, d\lambda\, d\rho, \tag{1}$$

where Θ is the 13-dimensional polytope $\Delta_1 \times \Delta_3 \times \Delta_3 \times \Delta_3 \times \Delta_3$. The factors are probability simplices,

$$\begin{aligned}
\Delta_1 &= \{(\pi,\tau) \in \mathbb{R}^2_{\geq 0} : \pi + \tau = 1\}, \\
\Delta_3 &= \{(\lambda_{\mathtt{A}}^{(k)}, \lambda_{\mathtt{C}}^{(k)}, \lambda_{\mathtt{G}}^{(k)}, \lambda_{\mathtt{T}}^{(k)}) \in \mathbb{R}^4_{\geq 0} \ : \textstyle\sum_i \lambda_i^{(k)} = 1\}, \quad k = 1,2, \\
\Delta_3 &= \{(\rho_{\mathtt{A}}^{(k)}, \rho_{\mathtt{C}}^{(k)}, \rho_{\mathtt{G}}^{(k)}, \rho_{\mathtt{T}}^{(k)}) \in \mathbb{R}^4_{\geq 0} \ : \textstyle\sum_i \rho_i^{(k)} = 1\}, \quad k = 1,2.
\end{aligned}$$

and we integrate with respect to Lebesgue probability measure on Θ. If we take the exponents U_{ij} to be the entries of the particular contingency table

$$U = \begin{pmatrix} 4 & 2 & 2 & 2 \\ 2 & 4 & 2 & 2 \\ 2 & 2 & 4 & 2 \\ 2 & 2 & 2 & 4 \end{pmatrix}, \tag{2}$$

then the exact value of the integral (1) is the rational number

$$\frac{571 \cdot 773426813 \cdot 17682039596993 \cdot 625015426432626533}{2^{31} \cdot 3^{20} \cdot 5^{12} \cdot 7^{11} \cdot 11^8 \cdot 13^7 \cdot 17^5 \cdot 19^5 \cdot 23^5 \cdot 29^3 \cdot 31^3 \cdot 37^3 \cdot 41^3 \cdot 43^2}. \tag{3}$$

The table (2) is taken from Example 1.3 of Pachter and Sturmfels (2005), where the integrand

$$\prod_{i,j \in \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}} \bigl(\pi \lambda_i^{(1)} \lambda_j^{(2)} + \tau \rho_i^{(1)} \rho_j^{(2)}\bigr)^{U_{ij}} \tag{4}$$

was studied using the EM algorithm, and the problem of validating its global maximum over Θ was raised. See Feinberg et al. (2007, §4.2) and Sturmfels (2008, §3) for further discussions. That optimization problem, which was widely known as the 100 *Swiss Francs problem*, has in the meantime been solved by Gao et al. (2008).

The main difficulty in performing computations such as (1) = (3) lies in the fact that the expansion of the integrand has many terms. A first naive upper bound on the number of monomials in the expansion of (4) would be

$$\prod_{i,j \in \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}} (U_{ij}+1) \quad = \quad 3^{12} \cdot 5^4 \ = \ 332,150,625.$$

However, the true number of monomials is only $3,892,097$, and we obtain the rational number (3) by summing the values of the corresponding integrals

$$\int_\Theta \pi^{a_1} \tau^{a_2} (\lambda^{(1)})^u (\lambda^{(2)})^v (\rho^{(1)})^w (\rho^{(2)})^x d\pi\, d\tau\, d\lambda\, d\rho \quad =$$
$$\frac{a_1! a_2!}{(a_1+a_2+1)!} \cdot \frac{3! \prod_i u_i!}{(\sum_i u_i + 3)!} \cdot \frac{3! \prod_i v_i!}{(\sum_i v_i + 3)!} \cdot \frac{3! \prod_i w_i!}{(\sum_i w_i + 3)!} \cdot \frac{3! \prod_i x_i!}{(\sum_i x_i + 3)!}.$$

The geometric idea behind our approach is that the Newton polytope of (4) is a *zonotope* and we are summing over its lattice points. Definitions for these geometric objects are given in Section 3.

This paper is organized as follows. In Section 2 we describe the class of algebraic statistical models to which our method applies, and we specify the problem. In Section 3 we examine the Newton zonotopes of mixture models, and we derive formulas for marginal likelihood evaluation using tools from geometric combinatorics. Our algorithms and their implementations are described

in detail in Section 4. Section 5 is concerned with applications in Bayesian statistics. We show how *Dirichlet priors* can be incorporated into our approach, we discuss the evaluation of *Bayes factors*, we compare our setup with that of Chickering and Heckerman (1997), and we illustrate the scope of our methods by computing an integral arising from a data set of Evans et al. (1989).

A preliminary draft version of the present article was published as Section 5.2 of the Oberwolfach lecture notes (Drton et al., 2009). We refer to that volume for further information on the use of computational algebra in Bayesian statistics.

2. Independence Models and their Mixtures

We consider a collection of discrete random variables

$$\begin{array}{cccc} X_1^{(1)}, & X_2^{(1)}, & \ldots, & X_{s_1}^{(1)}, \\ X_1^{(2)}, & X_2^{(2)}, & \ldots, & X_{s_2}^{(2)}, \\ \vdots & \vdots & \ddots & \vdots \\ X_1^{(k)}, & X_2^{(k)}, & \ldots, & X_{s_k}^{(k)}, \end{array}$$

where $X_1^{(i)},\ldots,X_{s_i}^{(i)}$ are identically distributed with values in $\{0,1,\ldots,t_i\}$. The independence model $\mathcal{M}$ for these variables is a toric model (Pachter and Sturmfels, 2005, §1.2) represented by an integer $d \times n$-matrix A with

$$d = t_1 + t_2 + \cdots + t_k + k \quad \text{and} \quad n = \prod_{i=1}^{k} (t_i+1)^{s_i}. \tag{5}$$

The columns of the matrix A are indexed by elements v of the state space

$$\{0,1,\ldots,t_1\}^{s_1} \times \{0,1,\ldots,t_2\}^{s_2} \times \cdots \times \{0,1,\ldots,t_k\}^{s_k}. \tag{6}$$

The rows of the matrix A are indexed by the model parameters, which are the d coordinates of the points $\theta = (\theta^{(1)},\theta^{(2)},\ldots,\theta^{(k)})$ in the polytope

$$P \;=\; \Delta_{t_1} \times \Delta_{t_2} \times \cdots \times \Delta_{t_k}, \tag{7}$$

and the model $\mathcal{M}$ is the subset of the simplex Δ_{n-1} given parametrically by

$$p_v \quad = \quad \text{Prob}\big(X_j^{(i)} = v_j^{(i)} \text{ for all } i,j\big) \quad = \quad \prod_{i=1}^{k} \prod_{j=1}^{s_i} \theta^{(i)}_{v_j^{(i)}}. \tag{8}$$

This is a monomial in d unknowns. The matrix A is defined by taking its column a_v to be the exponent vector of this monomial.

In algebraic geometry, the model $\mathcal{M}$ is known as *Segre-Veronese variety*

$$\mathbb{P}^{t_1} \times \mathbb{P}^{t_2} \times \cdots \times \mathbb{P}^{t_k} \quad \hookrightarrow \quad \mathbb{P}^{n-1}, \tag{9}$$

where the embedding is given by the line bundle $O(s_1,s_2,\ldots,s_k)$. The manifold $\mathcal{M}$ is the toric variety of the polytope P. Both objects have dimension $d-k$, and they are identified with each other via the moment map (Fulton, 1993, §4).

Example 1 *Consider three binary random variables where the last two random variables are identically distributed. In our notation, this corresponds to $k = 2$, $s_1 = 1$, $s_2 = 2$ and $t_1 = t_2 = 1$. We find that $d = 4, n = 8$, and*

$$A \quad = \quad \begin{array}{c} \\ \theta_0^{(1)} \\ \theta_1^{(1)} \\ \theta_0^{(2)} \\ \theta_1^{(2)} \end{array} \begin{array}{c} \begin{array}{cccccccc} p_{000} & p_{001} & p_{010} & p_{011} & p_{100} & p_{101} & p_{110} & p_{111} \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 0 & 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 2 & 0 & 1 & 1 & 2 \end{pmatrix} \end{array}.$$

The columns of this matrix represent the monomials in the parametrization (8). The model $\mathcal{M}$ lies in the 5-dimensional subsimplex of Δ_7 given by $p_{001} = p_{010}$ and $p_{101} = p_{110}$, and it consists of all rank one matrices

$$\begin{pmatrix} p_{000} & p_{001} & p_{100} & p_{101} \\ p_{010} & p_{011} & p_{110} & p_{111} \end{pmatrix}.$$

In algebraic geometry, the surface $\mathcal{M}$ is called a rational normal scroll.

The matrix A has repeated columns whenever $s_i \geq 2$ for some i. It is sometimes convenient to represent the model $\mathcal{M}$ by the matrix $\tilde{A}$ which is obtained from A by removing repeated columns. We label the columns of $\tilde{A}$ by elements $v = (v^{(1)}, \ldots, v^{(k)})$ of (6) whose components $v^{(i)} \in \{0, 1, \ldots, t_i\}^{s_i}$ are weakly increasing. Hence $\tilde{A}$ is a $d \times \tilde{n}$-matrix with

$$\tilde{n} \quad = \quad \prod_{i=1}^{k} \binom{s_i + t_i}{s_i}. \tag{10}$$

The model $\mathcal{M}$ and its mixtures are subsets of a subsimplex $\Delta_{\tilde{n}-1}$ of Δ_{n-1}.

We now introduce *marginal likelihood integrals*. All our domains of integration in this paper are polytopes that are products of standard probability simplices. On each such polytope we fix the standard Lebesgue probability measure. In other words, our discussion of Bayesian inference refers to the uniform prior on each parameter space. Naturally, other prior distributions, such as Dirichlet priors, are of interest, and our methods are extended to these in Section 5. In what follows, we simply work with uniform priors.

We identify the state space (6) with the set $\{1, \ldots, n\}$. A *data vector* $U = (U_1, \ldots, U_n)$ is thus an element of $\mathbb{N}^n$. The *sample size* of these data is $U_1 + U_2 + \cdots + U_n = N$. If the sample size N is fixed then the probability of observing these data is

$$\mathbf{L}_U(\theta) \quad = \quad \frac{N!}{U_1! U_2! \cdots U_n!} \cdot p_1(\theta)^{U_1} \cdot p_2(\theta)^{U_2} \cdot \cdots \cdot p_n(\theta)^{U_n}.$$

This expression is a function on the polytope P which is known as the *likelihood function* of the data U with respect to the independence model $\mathcal{M}$. The *marginal likelihood* of the data U with respect to the model $\mathcal{M}$ equals

$$\int_P \mathbf{L}_U(\theta)\, d\theta.$$

The value of this integral is a rational number which we now compute explicitly. The data U will enter this calculation by way of the *sufficient statistic* $b = A \cdot U$, which is a vector in $\mathbb{N}^d$. The

coordinates of this vector are denoted $b_j^{(i)}$ for $i = 1,\ldots,k$ and $j = 0,\ldots,t_k$. Thus $b_j^{(i)}$ is the total number of times the value j is attained by one of the random variables $X_1^{(i)},\ldots,X_{s_i}^{(i)}$ in the i-th group. Clearly, the sufficient statistics satisfy

$$b_0^{(i)} + b_1^{(i)} + \cdots + b_{t_i}^{(i)} \;=\; s_i \cdot N \qquad \text{for all } i = 1,2,\ldots,k. \tag{11}$$

The likelihood function $\mathbf{L}_U(\theta)$ is the constant $\frac{N!}{U_1!\cdots U_n!}$ times the monomial

$$\theta^b \;=\; \prod_{i=1}^{k}\prod_{j=0}^{t_i}(\theta_j^{(i)})^{b_j^{(i)}}.$$

The logarithm of this function is concave on the polytope P, and its maximum value is attained at the point $\hat{\theta}$ with coordinates $\hat{\theta}_j^{(i)} = b_j^{(i)}/(s_i \cdot N)$.

Lemma 1 *The integral of the monomial θ^b over the polytope P equals*

$$\int_P \theta^b d\theta \quad = \quad \prod_{i=1}^{k} \frac{t_i!\, b_0^{(i)}!\, b_1^{(i)}! \cdots b_{t_i}^{(i)}!}{(s_i N + t_i)!}.$$

The product of this number with the multinomial coefficient $N!/(U_1!\cdots U_n!)$ equals the marginal likelihood of the data U for the independence model $\mathcal{M}$.

Proof Since P is the product of simplices (7), this follows from the formula

$$\int_{\Delta_t} \theta_0^{b_0}\theta_1^{b_1}\cdots\theta_t^{b_t} d\theta \quad = \quad \frac{t!\cdot b_0!\cdot b_1!\cdots b_t!}{(b_0+b_1+\cdots+b_t+t)!} \tag{12}$$

for the integral of a monomial over the standard probability simplex Δ_t. ■

Our objective is to compute marginal likelihood integrals for the mixture model $\mathcal{M}^{(2)}$. The natural parameter space of this model is the polytope

$$\Theta \;=\; \Delta_1 \times P \times P.$$

Let $a_v \in \mathbb{N}^d$ be the column vector of A indexed by the state v, which is either in (6) or in $\{1,2,\ldots,n\}$. The parametrization (8) can be written simply as $p_v = \theta^{a_v}$. The mixture model $\mathcal{M}^{(2)}$ is defined to be the subset of Δ_{n-1} with the parametric representation

$$p_v \;=\; \sigma_0 \cdot \theta^{a_v} + \sigma_1 \cdot \rho^{a_v} \qquad \text{for } (\sigma,\theta,\rho) \in \Theta. \tag{13}$$

The likelihood function of a data vector $U \in \mathbb{N}^n$ for the model $\mathcal{M}^{(2)}$ equals

$$\mathbf{L}_U(\sigma,\theta,\rho) \quad = \quad \frac{N!}{U_1!U_2!\cdots U_n!}\, p_1(\sigma,\theta,\rho)^{U_1}\cdots p_n(\sigma,\theta,\rho)^{U_n}. \tag{14}$$

The *marginal likelihood* of the data U with respect to the model $\mathcal{M}^{(2)}$ equals

$$\int_\Theta \mathbf{L}_U(\sigma,\theta,\rho)\, d\sigma d\theta d\rho \;=\; \frac{N!}{U_1!\cdots U_n!}\int_\Theta \prod_v (\sigma_0\theta^{a_v} + \sigma_1\rho^{a_v})^{U_v} d\sigma d\theta d\rho. \tag{15}$$

The following proposition shows that we can evaluate this integral *exactly*.

Proposition 2 *The marginal likelihood (15) is a rational number.*

Proof The likelihood function $\mathbf{L}_U$ is a $\mathbb{Q}_{\geq 0}$-linear combination of monomials $\sigma^a\theta^b\rho^c$. The integral (15) is the same $\mathbb{Q}_{\geq 0}$-linear combination of the numbers

$$\int_\Theta \sigma^a\theta^b\rho^c d\sigma d\theta d\rho \quad = \quad \Big(\int_{\Delta_1}\sigma^a d\sigma\Big)\cdot\Big(\int_P \theta^b d\theta\Big)\cdot\Big(\int_P \rho^c d\rho\Big).$$

Each of the three factors is an easy-to-evaluate rational number, by (12). ■

Example 2 *The integral (1) expresses the marginal likelihood of a* 4×4*-table of counts* $U=(U_{ij})$ *with respect to the mixture model* $\mathcal{M}^{(2)}$*. Specifically, the marginal likelihood of the data (2) equals the normalizing constant* $40!\cdot(2!)^{-12}\cdot(4!)^{-4}$ *times the number (3). The model* $\mathcal{M}^{(2)}$ *consists of all non-negative* 4×4*-matrices of rank* ≤ 2 *whose entries sum to one. Here the parametrization (13) is not identifiable because* $\dim(\mathcal{M}^{(2)})=11$ *but* $\dim(\Theta)=13$*. In this example,* $k=2$*,* $s_1{=}s_2{=}1$*,* $t_1{=}t_2{=}3$*,* $d=8$*,* $n=16$*.*

In algebraic geometry, the model $\mathcal{M}^{(2)}$ is known as the first secant variety of the Segre-Veronese variety (9). We could also consider the higher secant varieties $\mathcal{M}^{(l)}$, which correspond to mixtures of l independent distributions, and much of our analysis can be extended to that case, but for simplicity we restrict ourselves to $l=2$. The variety $\mathcal{M}^{(2)}$ is embedded in the projective space $\mathbb{P}^{\tilde{n}-1}$ with $\tilde{n}$ as in (10). Note that $\tilde{n}$ can be much smaller than n. If this is the case, it is convenient to aggregate states whose probabilities are identical and represent the data by a vector $\tilde{U}\in\mathbb{N}^{\tilde{n}}$. Here is an example.

Example 3 *Let* $k{=}1$*,* $s_1{=}4$ *and* $t_1{=}1$*, so* $\mathcal{M}$ *is the independence model for four identically distributed binary random variables. Then* $d=2$ *and* $n=16$*. The corresponding integer matrix and its row and column labels are*

$$A \;=\; \begin{array}{c} \\ \theta_0 \\ \theta_1 \end{array}\!\!\begin{array}{c} \begin{array}{cccccccccc} p_{0000} & p_{0001} & p_{0010} & p_{0100} & p_{1000} & p_{0011} & \cdots & p_{1110} & p_{1111} \end{array} \\ \begin{pmatrix} 4 & 3 & 3 & 3 & 3 & 2 & \cdots & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 2 & \cdots & 3 & 4 \end{pmatrix} \end{array}.$$

However, this matrix has only $\tilde{n}=5$ *distinct columns, and we instead use*

$$\tilde{A} \;=\; \begin{array}{c} \\ \theta_0 \\ \theta_1 \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} p_0 & p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{pmatrix} 4 & 3 & 2 & 1 & 0 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix} \end{array}.$$

The mixture model $\mathcal{M}^{(2)}$ *is the subset of* Δ_4 *given by the parametrization*

$$p_i \quad = \quad \binom{4}{i}\cdot\big(\sigma_0\cdot\theta_0^{4-i}\cdot\theta_1^i \,+\, \sigma_1\cdot\rho_0^{4-i}\cdot\rho_1^i\big) \qquad \text{for } i=0,1,2,3,4.$$

In algebraic geometry, this threefold is the secant variety of the rational normal curve in $\mathbb{P}^4$*. This is the cubic hypersurface with the implicit equation*

$$\det\begin{bmatrix} 12p_0 & 3p_1 & 2p_2 \\ 3p_1 & 2p_2 & 3p_3 \\ 2p_2 & 3p_3 & 12p_4 \end{bmatrix} \quad = \quad 0.$$

In Hoşten et al. (2005, Example 9), the likelihood function (14) was studied for the data

$$\tilde{U} \quad = \quad (\tilde{U}_0, \tilde{U}_1, \tilde{U}_2, \tilde{U}_3, \tilde{U}_4) \quad = \quad (51, 18, 73, 25, 75).$$

It has three local maxima (modulo swapping θ *and* ρ*) whose coordinates are algebraic numbers of degree* 12*. Using the methods to be described in the next two sections, we computed the exact value of the marginal likelihood for the data* $\tilde{U}$ *with respect to* $\mathcal{M}^{(2)}$*. The rational number (15) is found to be the ratio of two relatively prime integers having* 530 *digits and* 552 *digits, and its numerical value is approximately* $0.77887163388386786113357 42 \cdot 10^{-22}$.

3. Summation over a Zonotope

Our starting point is the observation that the Newton polytope of the likelihood function (14) is a zonotope. Recall that the *Newton polytope* of a polynomial is the convex hull of all exponent vectors appearing in the expansion of that polynomial, and a polytope is a *zonotope* if it is the image of a standard cube under a linear map. See Cox et al. (2005, §7) and Ziegler (1995, §7) for further discussions. We are here considering the zonotope

$$Z_A(U) \;=\; \sum_{v=1}^{n} U_v \cdot [0, a_v],$$

where $[0, a_v]$ represents the line segment between the origin and the point $a_v \in \mathbb{R}^d$, and the sum is a Minkowski sum of line segments. We write $Z_A = Z_A(1, 1, \ldots, 1)$ for the basic zonotope spanned by the vectors a_v. Hence $Z_A(U)$ is obtained by stretching Z_A along those vectors by factors U_v respectively. Assuming that the counts U_v are all positive, we have

$$\dim(Z_A(U)) \;=\; \dim(Z_A) \;=\; \operatorname{rank}(A) \;=\; d-k+1. \tag{16}$$

The zonotope Z_A is related to the polytope $P = \operatorname{conv}(A)$ in (7) as follows. The dimension $d-k = t_1 + \cdots + t_k$ of P is one less than $\dim(Z_A)$, and P appears as the *vertex figure* of the zonotope Z_A at the distinguished vertex 0.

Remark 3 *For higher mixtures* $\mathcal{M}^{(l)}$*, the Newton polytope of the likelihood function is isomorphic to the Minkowski sum of* $(l-1)$*-dimensional simplices in* $\mathbb{R}^{(l-1)d}$*. Only when* $l = 2$*, this Minkowski sum is a zonotope.*

The marginal likelihood (15) we wish to compute is the integral

$$\int_{\Theta} \prod_{v=1}^{n} (\sigma_0 \theta^{a_v} + \sigma_1 \rho^{a_v})^{U_v} d\sigma d\theta d\rho \tag{17}$$

times the constant $N!/(U_1! \cdots U_n!)$. Our approach to this computation is to sum over the lattice points in the zonotope $Z_A(U)$. If the matrix A has repeated columns, we may replace A with the reduced matrix $\tilde{A}$ and U with the corresponding reduced data vector $\tilde{U}$. If one desires the marginal likelihood for the reduced data vector $\tilde{U}$ instead of the original data vector U, the integral remains the same while the normalizing constant becomes

$$\frac{N!}{\tilde{U}_1! \cdots \tilde{U}_{\tilde{n}}!} \cdot \alpha_1^{\tilde{U}_1} \cdots \alpha_{\tilde{n}}^{\tilde{U}_{\tilde{n}}},$$

where α_i is the number of columns in A equal to the i-th column of $\tilde{A}$. In what follows we ignore the normalizing constant and focus on computing the integral (17) with respect to the original matrix A.

For a vector $b \in \mathbb{R}^d_{\geq 0}$ we let $|b|$ denote its L^1-norm $\sum_{t=1}^d b_t$. Recall from (8) that all columns of the $d \times n$-matrix A have the same coordinate sum

$$a \ := \ |a_v| \ = \ s_1 + s_2 + \cdots + s_k, \ \text{ for all } v = 1,2,\ldots,n,$$

and from (11) that we may denote the entries of a vector $b \in \mathbb{R}^d$ by $b_j^{(i)}$ for $i = 1,\ldots,k$ and $j = 0,\ldots,t_k$. Also, let $\mathbb{L}$ denote the image of the linear map $A : \mathbb{Z}^n \to \mathbb{Z}^d$. Thus $\mathbb{L}$ is a sublattice of rank $d-k+1$ in $\mathbb{Z}^d$. We abbreviate $Z_A^{\mathbb{L}}(U) := Z_A(U) \cap \mathbb{L}$. Now, using the binomial theorem, we have

$$(\sigma_0\theta^{a_v} + \sigma_1\rho^{a_v})^{U_v} \ = \ \sum_{x_v=0}^{U_v} \binom{U_v}{x_v} \sigma_0^{x_v}\sigma_1^{U_v-x_v}\theta^{x_v \cdot a_v}\rho^{(U_v-x_v)\cdot a_v}.$$

Therefore, in the expansion of the integrand in (17), the exponents of θ are of the form of $b = \sum_v x_v a_v \in Z_A^{\mathbb{L}}(U)$, $0 \leq x_v \leq U_v$. The other exponents may be expressed in terms of b. This gives us

$$\prod_{v=1}^n (\sigma_0\theta^{a_v} + \sigma_1\rho^{a_v})^{U_v} \ = \sum_{\substack{b \in Z_A^{\mathbb{L}}(U) \\ c = AU-b}} \phi_A(b,U) \cdot \sigma_0^{|b|/a} \cdot \sigma_1^{|c|/a} \cdot \theta^b \cdot \rho^c. \tag{18}$$

Writing $\mathbf{D}(U) = \{(x_1,\ldots,x_n) \in \mathbb{Z}^n : 0 \leq x_v \leq U_v, v = 1,\ldots,n\}$, the coefficient in (18) equals

$$\phi_A(b,U) \ = \sum_{\substack{Ax=b \\ x \in \mathbf{D}(U)}} \prod_{v=1}^n \binom{U_v}{x_v}. \tag{19}$$

Thus, by formulas (12) and (18), the integral (17) evaluates to

$$\sum_{\substack{b \in Z_A^{\mathbb{L}}(U) \\ c = AU-b}} \phi_A(b,U) \cdot \frac{(|b|/a)!\,(|c|/a)!}{(|U|+1)!} \cdot \prod_{i=1}^k \left(\frac{t_i!\, b_0^{(i)}! \cdots b_{t_i}^{(i)}!}{(|b^{(i)}|+t_i)!} \, \frac{t_i!\, c_0^{(i)}! \cdots c_{t_i}^{(i)}!}{(|c^{(i)}|+t_i)!} \right). \tag{20}$$

We summarize the result of this derivation in the following theorem.

Theorem 4 *The marginal likelihood of the data U in the mixture model $\mathcal{M}^{(2)}$ is equal to the sum (20) times the normalizing constant $N!/(U_1!\cdots U_n!)$.*

Each individual summand in the formula (20) is a ratio of factorials and hence can be evaluated symbolically. The challenge in turning Theorem 4 into a practical algorithm lies in the fact that both of the sums (19) and (20) are over very large sets. We shall discuss these challenges and present techniques from both computer science and mathematics for addressing them.

We first turn our attention to the coefficients $\phi_A(b,U)$ of the expansion (18). These quantities are written as an explicit sum in (19). The first useful observation is that these coefficients are also the coefficients of the expansion

$$\prod_v (\theta^{a_v} + 1)^{U_v} \quad = \sum_{b \in Z_A^{\mathbb{L}}(U)} \phi_A(b,U) \cdot \theta^b, \tag{21}$$

which comes from substituting $\sigma_i = 1$ and $\rho_j = 1$ in (18). When the cardinality of $Z_A^{\mathbb{L}}(U)$ is sufficiently small, the quantity $\phi_A(b,U)$ can be computed quickly by expanding (21) using a computer algebra system. We used MAPLE for this and all other symbolic computations in this project.

If the expansion (21) is not feasible, then it is tempting to compute the individual $\phi_A(b,U)$ via the sum-product formula (19). This method requires summation over the set $\{x \in \mathbf{D}(U) : Ax = b\}$, which is the set of lattice points in an $(n-d+k-1)$-dimensional polytope. Even if this loop can be implemented, performing the sum in (19) symbolically requires the evaluation of many large binomials, causing the process to be rather inefficient.

An alternative is offered by the following recurrence formula:

$$\phi_A(b,U) \;=\; \sum_{x_n=0}^{U_n} \binom{U_n}{x_n} \phi_{A\setminus a_n}(b - x_n a_n, U \setminus U_n). \tag{22}$$

This is equivalent to writing the integrand in (17) as

$$\left(\prod_{v=1}^{n-1} (\sigma_0 \theta^{a_v} + \sigma_1 \rho^{a_v})^{U_v} \right) (\sigma_0 \theta^{a_n} + \sigma_1 \rho^{a_n})^{U_n}.$$

More generally, for each $0 < i < n$, we have the recurrence

$$\phi_A(b,U) = \sum_{b' \in Z_{A'}^{\mathbb{L}}(U')} \phi_{A'}(b',U') \cdot \phi_{A\setminus A'}(b-b', U \setminus U'),$$

where A' and U' consist of the first i columns and entries of A and U respectively. This corresponds to the factorization

$$\left(\prod_{v=1}^{i} (\sigma_0 \theta^{a_v} + \sigma_1 \rho^{a_v})^{U_v} \right) \left(\prod_{v=i+1}^{n} (\sigma_0 \theta^{a_v} + \sigma_1 \rho^{a_v})^{U_v} \right).$$

This formula gives flexibility in designing algorithms with different payoffs in time and space complexity, to be discussed in Section 4.

The next result records useful facts about the quantities $\phi_A(b,U)$.

Proposition 5 *Suppose $b \in \mathbb{Z}_A^{\mathbb{L}}(U)$ and $c = AU - b$. Then, the following quantities are all equal to $\phi_A(b,U)$:*
(1) $\#\{z \in \{0,1\}^N : A^U z = b\}$, where A^U is the extended matrix

$$A^U \;:=\; (\underbrace{a_1,\ldots,a_1}_{U_1}, \underbrace{a_2,\ldots,a_2}_{U_2}, \ldots, \underbrace{a_n,\ldots,a_n}_{U_n}),$$

(2) $\phi_A(c,U)$,
(3)

$$\sum_{\substack{Ax=b \\ l_j \le x_j \le u_j}} \prod_{v=1}^{n} \binom{U_v}{x_v},$$

where $u_j = \min\{U_j\} \cup \{b_m/a_{jm}\}_{m=1}^n$ and $l_j = U_j - \min\{U_j\} \cup \{c_m/a_{jm}\}_{m=1}^n$.

Proof (1) This follows directly from (21).
(2) For each $z \in \{0,1\}^N$ satisfying $A^U z = b$, note that $\bar{z} = (1,1,\ldots,1) - z$ satisfies $A^U \bar{z} = c$, and vice versa. The conclusion thus follows from (1).
(3) We require $Ax = b$ and $x \in \mathbf{D}(U)$. If $x_j > u_j = b_m/a_{jm}$ then $a_{jm}x_j > b_m$, which implies $Ax \neq b$. The lower bound is derived by a similar argument. ■

One aspect of our approach is the decision, for any given model A and data set U, whether or not to attempt the expansion (21) using computer algebra. This decision depends on the cardinality of the set $Z_A^{\mathbb{L}}(U)$. In what follows, we compute the number exactly when A is unimodular. When A is not unimodular, we obtain useful lower and upper bounds.

Let S be any subset of the columns of A. We call S *independent* if its elements are linearly independent in $\mathbb{R}^d$. With S we associate the integer

$$\text{index}(S) \;:=\; [\mathbb{R}S \cap \mathbb{L} : \mathbb{Z}S].$$

This is the index of the abelian group generated by S inside the possibly larger abelian group of all lattice points in $\mathbb{L} = \mathbb{Z}A$ that lie in the span of S. The following formula is due to R. Stanley and appears in Stanley (1991, Theorem 2.2):

Proposition 6 *The number of lattice points in the zonotope $Z_A(U)$ equals*

$$\#Z_A^{\mathbb{L}}(U) \;=\; \sum_{S \subseteq A \text{ indep.}} \text{index}(S) \cdot \prod_{a_v \in S} U_v. \tag{23}$$

In fact, the number of monomials in (18) equals $\#M_A(U)$, where $M_A(U)$ is the set $\{b \in Z_A^{\mathbb{L}}(U) : \phi_A(b,U) \neq 0\}$, and this set can be different from $Z_A^{\mathbb{L}}(U)$. For that number we have the following bounds. The proof, which uses the methods in Stanley (1991, §2), will be omitted here.

Theorem 7 *The number $\#M_A(U)$ of monomials in the expansion (18) of the likelihood function to be integrated satisfies the two inequalities*

$$\sum_{S \subseteq A \text{ indep.}} \prod_{v \in S} U_v \;\leq\; \#M_A(U) \;\leq\; \sum_{S \subseteq A \text{ indep.}} \text{index}(S) \cdot \prod_{v \in S} U_v. \tag{24}$$

By definition, the matrix A is *unimodular* if $\text{index}(S) = 1$ for all independent subsets S of the columns of A. In this case, the upper bound coincides with the lower bound, and so $M_A(U) = Z_A^{\mathbb{L}}(U)$. This happens in the classical case of two-dimensional contingency tables ($k = 2$ and $s_1 = s_2 = 1$). In general, $\#Z_A^{\mathbb{L}}(U)/\#M_A(U)$ tends to 1 when all coordinates of U tend to infinity. This is why we believe that for computational purposes, $\#Z_A^{\mathbb{L}}(U)$ is a good approximation of $\#M_A(U)$.

Remark 8 *There exist integer matrices A for which $\#M_A(U)$ does not agree with the upper bound in Theorem 7. However, we conjecture that $\#M_A(U) = \#Z_A^{\mathbb{L}}(U)$ holds for matrices A of Segre-Veronese type as in (8) and strictly positive data vectors U.*

Example 4 *Consider the* 100 Swiss Francs *example in Section 1. Here A is unimodular and it has* 16145 *independent subsets S. The corresponding sum of* 16145 *squarefree monomials in (23) gives the number of terms in the expansion of (4). For the data U in (2) this sum evaluates to* $3,892,097$.

Example 5 *We consider the matrix and data from Example 3.*

$$\begin{aligned} \tilde{A} &= \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 & 0 \end{pmatrix} \\ \tilde{U} &= (51, 18, 73, 25, 75) \end{aligned}$$

By Theorem 7, the lower bound is 22,273 and the upper bound is 48,646. Here the number $\#M_{\tilde{A}}(\tilde{U})$ *of monomials agrees with the latter.*

We next present a formula for $\text{index}(S)$ when S is any linearly independent subset of the columns of the matrix A. After relabeling we may assume that $S = \{a_1, \ldots, a_k\}$ consists of the first k columns of A. Let $H = VA$ denote the row Hermite normal form of A. Here $V \in SL_d(\mathbb{Z})$ and H satisfies

$$H_{ij} = 0 \text{ for } i > j \text{ and } 0 \leq H_{ij} < H_{jj} \text{ for } i < j.$$

Hermite normal form is a built-in function in computer algebra systems. For instance, in MAPLE the command is `ihermite`. Using the invertible matrix V, we may replace A with H, so that $\mathbb{R}S$ becomes $\mathbb{R}^k$ and $\mathbb{Z}S$ is the image over $\mathbb{Z}$ of the upper left $k \times k$-submatrix of H. We seek the index of that lattice in the possibly larger lattice $\mathbb{Z}A \cap \mathbb{Z}^k$. To this end we compute the column Hermite normal form $H' = HV'$. Here $V' \in SL_n(\mathbb{Z})$ and H' satisfies

$$H'_{ij} = 0 \text{ if } i > j \text{ or } j > d \text{ and } 0 \leq H_{ij} < H_{ii} \text{ for } i < j.$$

The lattice $\mathbb{Z}A \cap \mathbb{Z}^k$ is spanned by the first k columns of H', and this implies

$$\text{index}(S) \quad = \quad \frac{H_{11} H_{22} \cdots H_{kk}}{H'_{11} H'_{22} \cdots H'_{kk}}.$$

4. Algorithms

In this section we discuss algorithms for computing the integral (17) exactly, and we discuss their advantages and limitations. In particular, we examine four main techniques which represent the formulas (20), (21), (16) and (22) respectively. The practical performance of the various algorithms is compared by computing the integral in Example 3.

A MAPLE library which implements our algorithms is made available at

`http://math.berkeley.edu/~shaowei/integrals.html`.

The input for our MAPLE code consists of parameter vectors $s = (s_1, \ldots, s_k)$ and $t = (t_1, \ldots, t_k)$ as well as a data vector $U \in \mathbb{N}^n$. This input uniquely specifies the $d \times n$-matrix A. Here d and n are as in (5). The output features the matrices A and $\tilde{A}$, the marginal likelihood integrals for $\mathcal{M}$ and $\mathcal{M}^{(2)}$, as well as the bounds in (24).

We tacitly assume that A has been replaced with the reduced matrix $\tilde{A}$. Thus from now on we assume that A has no repeated columns. This requires some care concerning the normalizing constants. All columns of the matrix A have the same coordinate sum a, and the convex hull of the columns is the polytope $P = \Delta_{t_1} \times \Delta_{t_2} \times \cdots \times \Delta_{t_k}$. Our domain of integration is the following polytope of dimension $2d - 2k + 1$:

$$\Theta \;=\; \Delta_1 \times P \times P.$$

We seek to compute the rational number

$$\int_{\Theta} \prod_{v=1}^{n} (\sigma_0 \theta^{a_v} + \sigma_1 \rho^{a_v})^{U_v} d\sigma d\theta d\rho, \tag{25}$$

where integration is with respect to Lebesgue probability measure. Our MAPLE code outputs this integral multiplied with the statistically correct normalizing constant. That constant will be ignored in what follows. In our complexity analysis, we fix A while allowing the data U to vary. The complexities will be given in terms of the sample size $N = U_1 + \cdots + U_n$.

4.1 Ignorance is Costly

Given an integration problem such as (25), a first attempt is to use the symbolic integration capabilities of a computer algebra package such as MAPLE. We refer to this method as *ignorant integration*:

```
U  := [51, 18, 73, 25, 75]:
f  := (s*t^4        +(1-s)*p^4        )^U[1] *
      (s*t^3*(1-t)  +(1-s)*p^3*(1-p)  )^U[2] *
      (s*t^2*(1-t)^2+(1-s)*p^2*(1-p)^2)^U[3] *
      (s*t  *(1-t)^3+(1-s)*p  *(1-p)^3)^U[4] *
      (s    *(1-t)^4+(1-s)    *(1-p)^4)^U[5]:
II := int(int(int(f,p=0..1),t=0..1),s=0..1);
```

In the case of mixture models, recognizing the integral as the sum of integrals of monomials over a polytope allows us to avoid the expensive integration step above by using (20). To demonstrate the power of using (20), we implemented a simple algorithm that computes each $\phi_A(b,U)$ using the naive expansion in (19). We computed the integral in Example 3 with a small data vector $U = (2,2,2,2,2)$, which is the rational number

$$\frac{66364720654753}{5905738398721701533994000},$$

and summarize the run-times and memory usages of the two algorithms in the table below. All experiments reported in this section are done in MAPLE.

	Time(seconds)	Memory(bytes)
Ignorant Integration	16.331	155,947,120
Naive Expansion	0.007	458,668

For the remaining comparisons in this section, we no longer consider the ignorant integration algorithm because it is computationally too expensive.

4.2 Symbolic Expansion of the Integrand

While ignorant use of a computer algebra system is unsuitable for computing our integrals, we can still exploit its powerful polynomial expansion capabilities to find the coefficients of (21). A major advantage is that it is very easy to write code for this method. We compare the performance of this symbolic expansion algorithm against that of the naive expansion algorithm. The table below concerns computing the coefficients $\phi_A(b,U)$ for the original data $U = (51,18,73,25,75)$. The

column "Extract" refers to the time taken to extract the coefficients $\phi_A(b,U)$ from the expansion of the polynomial, while the column "Sum" shows the time taken to evaluate (20) after all the needed values of $\phi_A(b,U)$ had been computed and extracted.

	Time(seconds)				Memory
	$\phi_A(b,U)$	Extract	Sum	Total	(bytes)
Naive Expansion	2764.35	-	31.19	2795.54	10,287,268
Symbolic Expansion	28.73	962.86	29.44	1021.03	66,965,528

4.3 Storage and Evaluation of $\phi_A(b,U)$

Symbolic expansion is fast for computing $\phi_A(b,U)$, but it has two drawbacks: high memory usage and the long time it takes to extract the values of $\phi_A(b,U)$. One solution is to create specialized data structures and algorithms for expanding (21), rather using than those offered by MAPLE.

First, we tackle the problem of storing the coefficients $\phi_A(b,U)$ for $b \in Z_A^{\mathbb{L}}(U) \subset \mathbb{R}^d$ as they are being computed. One naive method is to use a d-dimensional array $\phi[\cdot]$. However, noting that A is not row rank full, we can use a d_0-dimensional array to store $\phi_A(b,U)$, where $d_0 = \text{rank}(A) = d-k+1$. Furthermore, by Proposition 5(2), the expanded integrand is a symmetric polynomial, so only half the coefficients need to be stored. We will leave out the implementation details so as not to complicate our discussions. In our algorithms, we will assume that the coefficients are stored in a d_0-dimensional array $\phi[\cdot]$, and the entry that represents $\phi_A(b,U)$ will be referred to as $\phi[b]$.

Next, we discuss how $\phi_A(b,U)$ can be computed. One could use the naive expansion (19), but this involves evaluating many binomials coefficients and products, so the algorithm is inefficient for data vectors with large coordinates. A more efficient solution uses the recurrence formula (22):

Algorithm 1 (RECURRENCE(A, U))
Input: *The matrix A and the vector* U.
Output: *The coefficients* $\phi_A(b,U)$.
Step 1*: Create a* d_0*-dimensional array* ϕ *of zeros.*
Step 2*: For each* $x \in \{0,1,\ldots,U_1\}$ *set*

$$\phi[xa_1] := \binom{U_1}{x}.$$

Step 3*: Create a new* d_0*-dimensional array* ϕ'.
Step 4*: For each* $2 \le j \le n$ *do*
1. Set all the entries of ϕ' *to* 0.
2. For each $x \in \{0,1,\ldots,U_j\}$ *do*
For each non-zero entry $\phi[b]$ *in* ϕ *do*
Increment $\phi'[b+xa_j]$ *by* $\binom{U_j}{x}\phi[b]$.
3. Replace ϕ *with* ϕ'.
Step 5*: Output the array* ϕ.

The space complexity of this algorithm is $O(N^{d_0})$ and its time complexity is $O(N^{d_0+1})$. By comparison, the naive expansion algorithm has space complexity $O(N^d)$ and time complexity $O(N^{n+1})$.

We now turn our attention to computing the integral (25). One major issue is the lack of memory to store all the terms of the expansion of the integrand. We overcome this problem by writing

the integrand as a product of smaller factors which can be expanded separately. In particular, we partition the columns of A into submatrices $A^{[1]},\ldots,A^{[m]}$ and let $U^{[1]},\ldots,U^{[m]}$ be the corresponding partition of U. Thus the integrand becomes

$$\prod_{j=1}^{m}\prod_{v}(\sigma_0\theta^{a_v^{[j]}}+\sigma_1\rho^{a_v^{[j]}})^{U_v^{[j]}},$$

where $a_v^{[j]}$ is the vth column in $A^{[j]}$. The resulting algorithm for evaluating the integral is as follows:

Algorithm 2 (Fast Integral)
Input: *The matrices $A^{[1]},\ldots,A^{[m]}$, vectors $U^{[1]},\ldots,U^{[m]}$ and the vector t.*
Output: *The value of the integral (25) in exact rational arithmetic.*
Step 1*: For $1\le j\le m$, compute $\phi^{[j]}:=\mathrm{RECURRENCE}(A^{[j]},U^{[j]})$.*
Step 2*: Set $I:=0$.*
Step 3*: For each non-zero entry $\phi^{[1]}[b^{[1]}]$ in $\phi^{[1]}$ do*

$$\vdots$$

For each non-zero entry $\phi^{[m]}[b^{[m]}]$ in $\phi^{[m]}$ do
Set $b:=b^{[1]}+\cdots+b^{[m]}$, $c:=AU-b$, $\phi:=\prod_{j=1}^{m}\phi^{[j]}[b^{[j]}]$.
Increment I by

$$\phi\cdot\frac{(|b|/a)!(|c|/a)!}{(|U|+1)!}\cdot\prod_{i=1}^{k}\frac{t_i!b_0^{(i)}!\cdots b_{t_i}^{(i)}!}{(|b^{(i)}|+t_i)!}\,\frac{t_i!c_0^{(i)}!\cdots c_{t_i}^{(i)}!}{(|c^{(i)}|+t_i)!}.$$

Step 4*: Output the sum I.*

The algorithm can be sped up by precomputing the factorials used in the product in Step 3. The space and time complexity of this algorithm is $O(N^S)$ and $O(N^T)$ respectively, where $S=\max_i \mathrm{rank}\,A^{[i]}$ and $T=\sum_i \mathrm{rank}A^{[i]}$. From this, we see that the splitting of the integrand should be chosen wisely to achieve a good pay-off between the two complexities.

In the table below, we compare the naive expansion algorithm and the fast integral algorithm for the data $U=(51,18,73,25,75)$. We also compare the effect of splitting the integrand into two factors, as denoted by $m=1$ and $m=2$. For $m=1$, the fast integral algorithm takes significantly less time than naive expansion, and requires only about 1.5 times more memory.

	Time(minutes)	Memory(bytes)
Naive Expansion	43.67	9,173,360
Fast Integral (m=1)	1.76	13,497,944
Fast Integral (m=2)	139.47	6,355,828

4.4 Limitations and Applications

While our algorithms are optimized for exact evaluation of integrals for mixtures of independence models, they may not be practical for applications involving large sample sizes. To demonstrate their limitations, we vary the sample sizes in Example 3 and compare the computation times. The data vectors U are generated by scaling $U=(51,18,73,25,75)$ according to the sample size N and rounding off the entries. Here, N is varied from 110 to 300 by increments of 10. Figure 1 shows a logarithmic plot of the results. The times taken for $N=110$ and $N=300$ are 3.3 and 98.2 seconds

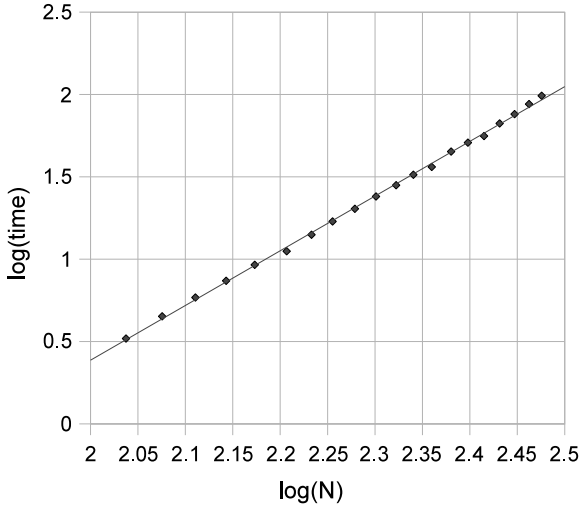


Figure 1: Comparison of computation time against sample size.

respectively. Computation times for larger samples may be extrapolated from the graph. Indeed, a sample size of 5000 could take more than 13 days.

For other models, such as the 100 *Swiss Francs* example in Section 1 and that of the schizophrenic patients in Example 9, the limitations are even more apparent. In the table below, for each example we list the sample size, computation time, rank of the corresponding A-matrix and the number of terms in the expansion of the integrand. Despite having smaller sample sizes, the computations for the latter two examples take a lot more time. This may be attributed to the higher ranks of the A-matrices and the larger number of terms that need to be summed up in our algorithm.

	Size	Time	Rank	#Terms
Coin Toss	242	45 sec	2	48,646
100 Swiss Francs	40	15 hrs	7	3,892,097
Schizophrenic Patients	132	16 days	5	34,177,836

Despite their high complexities, we believe our algorithms are important because they provide a gold standard with which approximation methods such as those studied in Chickering and Heckerman (1997) can be compared. Below, we use our exact methods to ascertain the accuracy of asymptotic formula derived in Watanabe (2001) and Watanabe and Yamazaki (2003, 2004) using desingularization methods from algebraic geometry.

Example 6 *Consider the model from Example 3. Choose data vectors* $U = (U_0, U_1, U_2, U_3, U_4)$ *with* $U_i = Nq_i$ *where* N *is a multiple of* 16 *and*

$$q_i = \frac{1}{16}\binom{4}{i}, \quad i = 0, 1, \ldots, 4.$$

Let $I_N(U)$ *be the integral (25). Define*

$$F_N(U) = N \sum_{i=0}^{4} q_i \log q_i - \log I_N(U).$$

According to Watanabe and Yamazaki (2004), for large N we have the asymptotics

$$\mathrm{E}_U[F_N(U)] \;=\; \frac{3}{4}\log N + O(1) \tag{26}$$

where the expectation E_U is taken over all U with sample size N under the distribution defined by $q = (q_0, q_1, q_2, q_3, q_4)$. Thus, we should expect

$$F_{16+N} - F_N \;\approx\; \frac{3}{4}\log(16+N) - \frac{3}{4}\log N =: g(N).$$

We compute $F_{16+N} - F_N$ using our exact methods and list the results below.

N	$F_{16+N} - F_N$	$g(N)$
16	*0.21027043*	*0.225772497*
32	*0.12553837*	*0.132068444*
48	*0.08977938*	*0.093704053*
64	*0.06993586*	*0.072682510*
80	*0.05729553*	*0.059385934*
96	*0.04853292*	*0.050210092*
112	*0.04209916*	*0.043493960*

Clearly, the table supports our conclusion. The coefficient $3/4$ of $\log N$ in the formula (26) is known as the real log-canonical threshold *of the statistical model. The example suggests that our method could be developed into a numerical technique for computing the real log-canonical threshold.*

5. Back to Bayesian Statistics

In this section we discuss how the exact integration approach presented here interfaces with issues in Bayesian statistics. The first concerns the rather restrictive assumption that our marginal likelihood integral be evaluated with respect to the uniform distribution (Lesbegue measure) on the parameter space Θ. It is standard practice to compute such integrals with respect to *Dirichlet priors*, and we shall now explain how our algorithms can be extended to Dirichlet priors. That extension is also available as a feature in our MAPLE implementation.

Recall that the *Dirichlet distribution* $\mathrm{Dir}(\alpha)$ is a continuous probability distribution parametrized by a vector $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_m)$ of positive reals. It is the multivariate generalization of the beta distribution and is conjugate prior (in the Bayesian sense) to the multinomial distribution. This means that the probability distribution function of $\mathrm{Dir}(\alpha)$ specifies the belief that the probability of the ith among $m+1$ events equals θ_i given that it has been observed $\alpha_i - 1$ times. More precisely, the probability density function $f(\theta;\alpha)$ of $\mathrm{Dir}(\alpha)$ is supported on the m-dimensional simplex

$$\Delta_m \;=\; \bigl\{(\theta_0,\ldots,\theta_m) \in \mathbb{R}^m_{\geq 0} \,:\, \theta_0 + \cdots + \theta_m = 1\bigr\},$$

and it equals

$$f(\theta_0,\ldots,\theta_m;\alpha_0,\ldots,\alpha_m) \quad = \quad \frac{1}{\mathbb{B}(\alpha)} \cdot \theta_0^{\alpha_0-1}\theta_1^{\alpha_1-1}\cdots\theta_m^{\alpha_m-1} \;=:\; \frac{\theta^{\alpha-1}}{\mathbb{B}(\alpha)}.$$

Here the normalizing constant is the multinomial beta function

$$\mathbb{B}(\alpha) \quad = \quad \frac{m!\Gamma(\alpha_0)\Gamma(\alpha_1)\cdots\Gamma(\alpha_m)}{\Gamma(\alpha_0+\alpha_1+\cdots+\alpha_m)}.$$

Note that, if the α_i are all integers, then this is the rational number

$$\mathbb{B}(\alpha) \;=\; \frac{m!(\alpha_0-1)!(\alpha_1-1)!\cdots(\alpha_m-1)!}{(\alpha_0+\cdots+\alpha_m-1)!}.$$

Thus the identity (12) is the special case of the identity $\int_{\Delta_m} f(\theta;\alpha)d\theta = 1$ for the density of the Dirichlet distribution when all $\alpha_i = b_i+1$ are integers.

We now return to the marginal likelihood for mixtures of independence models. To compute this quantity with respect to Dirichlet priors means the following. We fix positive real numbers α_0,α_1, and $\beta_j^{(i)}$ and $\gamma_j^{(i)}$ for $i=1,\ldots,k$ and $j=0,\ldots,t_i$. These specify Dirichlet distributions on Δ_1, P and P. Namely, the Dirichlet distribution on P given by the $\beta_j^{(i)}$ is the product probability measure given by taking the Dirichlet distribution with parameters $(\beta_0^{(i)},\beta_1^{(i)},\ldots,\beta_{t_i}^{(i)})$ on the i-th factor Δ_{t_i} in the product (7) and similarly for the $\gamma_j^{(i)}$. The resulting product probability distribution on $\Theta = \Delta_1\times P\times P$ is called the *Dirichlet distribution* with parameters (α,β,γ). Its probability density function is the product of the respective densities:

$$f(\sigma,\theta,\rho;\alpha,\beta,\gamma) \;=\; \frac{\sigma^{\alpha-1}}{\mathbb{B}(\alpha)}\cdot\prod_{i=1}^{k}\frac{(\theta^{(i)})^{\beta^{(i)}-1}}{\mathbb{B}(\beta^{(i)})}\cdot\prod_{i=1}^{k}\frac{(\rho^{(i)})^{\gamma^{(i)}-1}}{\mathbb{B}(\gamma^{(i)})}. \tag{27}$$

By the marginal likelihood with Dirichlet priors we mean the integral

$$\int_\Theta \mathbf{L}_U(\sigma,\theta,\rho)\, f(\sigma,\theta,\rho;\alpha,\beta,\gamma)d\sigma d\theta d\rho. \tag{28}$$

This is a modification of (15) and it depends not just on the data U and the model $\mathcal{M}^{(2)}$ but also on the choice of Dirichlet parameters (α,β,γ). When the coordinates of these parameters are arbitrary positive reals but not integers, then the value of the integral (28) is no longer a rational number. Nonetheless, it can be computed exactly as follows. We abbreviate the product of gamma functions in the denominator of the density (27) as follows:

$$\mathbb{B}(\alpha,\beta,\gamma) \;:=\; \mathbb{B}(\alpha)\cdot\prod_{i=1}^{k}\mathbb{B}(\beta^{(i)})\cdot\prod_{i=1}^{k}\mathbb{B}(\gamma^{(i)}).$$

Instead of the integrand (18) we now need to integrate

$$\sum_{\substack{b\in Z_A^{\mathbb{L}}(U)\\ c=AU-b}} \frac{\phi_A(b,U)}{\mathbb{B}(\alpha,\beta,\gamma)}\cdot\sigma_0^{|b|/a+\alpha_0-1}\cdot\sigma_1^{|c|/a+\alpha_1-1}\cdot\theta^{b+\beta-1}\cdot\rho^{c+\gamma-1}$$

with respect to Lebesgue probability measure on Θ. Doing this term by term, as before, we obtain the following modification of Theorem 4.

Corollary 9 *The marginal likelihood of the data U in the mixture model $\mathcal{M}^{(2)}$ with respect to Dirichlet priors with parameters (α,β,γ) equals*

$$\frac{N!}{U_1!\cdots U_n!\cdot\mathbb{B}(\alpha,\beta,\gamma)}\cdot\sum_{\substack{b\in Z_A^{\mathbb{L}}(U)\\ c=AU-b}}\phi_A(b,U)\,\frac{\Gamma(|b|/a+\alpha_0)\Gamma(|c|/a+\alpha_1)}{\Gamma(|U|+|\alpha|)}$$
$$\cdot\prod_{i=1}^{k}\Big(\frac{t_i!\Gamma(b_0^{(i)}+\beta_0^{(i)})\cdots\Gamma(b_{t_i}^{(i)}+\beta_{t_i}^{(i)})}{\Gamma(|b^{(i)}|+|\beta^{(i)}|)}\;\frac{t_i!\Gamma(c_0^{(i)}+\gamma_0^{(i)})\cdots\Gamma(c_{t_i}^{(i)}+\gamma_{t_i}^{(i)})}{\Gamma(|c^{(i)}|+|\gamma^{(i)}|)}\Big).$$

A well-known experimental study (Chickering and Heckerman, 1997) compares different methods for computing numerical approximations of marginal likelihood integrals. The model considered in the study is the *naive-Bayes model*, which, in the language of algebraic geometry, corresponds to arbitrary secant varieties of Segre varieties. In this paper we considered the first secant variety of arbitrary Segre-Veronese varieties. In what follows we restrict our discussion to the intersection of both classes of models, namely, to the first secant variety of Segre varieties. For the remainder of this section we fix

$$s_1 = s_2 = \cdots = s_k = 1$$

but we allow $t_1, t_2, \ldots, t_k$ to be arbitrary positive integers. Thus in the model of Chickering and Heckerman (1997, Equation 1), we fix $r_C = 2$, and the n there corresponds to our k.

To keep things as simple as possible, we shall fix the uniform distribution as in Sections 1–4 above. Thus, in the notation of Chickering and Heckerman (1997, §2), all Dirichlet hyperparameters α_{ijk} are set to 1. This implies that, for any data $U \in \mathbb{N}^n$ and any of our models, the problem of finding the maximum a posteriori (MAP) configuration is equivalent to finding the maximum likelihood (ML) configuration. To be precise, the *MAP configuration* is the point $(\hat{\sigma}, \hat{\theta}, \hat{\rho})$ in Θ which maximizes the likelihood function $\mathbf{L}_U(\sigma, \theta, \rho)$ in (14). This maximum may not be unique, and there will typically be many local maxima. Chickering and Heckerman (1997, §3.2) used the expectation maximization (EM) algorithm (Pachter and Sturmfels, 2005, §1.3) to approximate the MAP configuration numerically

The Laplace approximation and the BIC score (Chickering and Heckerman, 1997, §3.1) are predicated on the idea that the MAP configuration can be found with high accuracy and that the data U were actually drawn from the corresponding distribution $p(\hat{\sigma}, \hat{\theta}, \hat{\rho})$. Let $\mathbf{H}(\sigma, \theta, \rho)$ denote the Hessian matrix of the log-likelihood function $\log \mathbf{L}(\sigma, \theta, \rho)$. Then the Laplace approximation (Chickering and Heckerman, 1997, Equation 15) states that the logarithm of the marginal likelihood can be approximated by

$$\log \mathbf{L}(\hat{\sigma}, \hat{\theta}, \hat{\rho}) - \frac{1}{2} \log |\det \mathbf{H}(\hat{\sigma}, \hat{\theta}, \hat{\rho})| + \frac{2d - 2k + 1}{2} \log(2\pi). \tag{29}$$

The Bayesian information criterion (BIC) suggests the coarser approximation

$$\log \mathbf{L}(\hat{\sigma}, \hat{\theta}, \hat{\rho}) \; - \; \frac{2d - 2k + 1}{2} \log(N), \tag{30}$$

where $N = U_1 + \cdots + U_n$ is the sample size.

In algebraic statistics, we do not content ourselves with the output of the EM algorithm but, to the extent possible, we seek to actually solve the likelihood equations (Hoşten et al., 2005) and compute all local maxima of the likelihood function. We consider it a difficult problem to reliably find $(\hat{\sigma}, \hat{\theta}, \hat{\rho})$, and we are concerned about the accuracy of any approximation like (29) or (30).

Example 7 *Consider the* 100 Swiss Francs *table (2) discussed in the Introduction. Here* $k = 2$, $s_1 = s_2 = 1$, $t_1 = t_2 = 3$, *the matrix A is unimodular, and (9) is the Segre embedding* $\mathbb{P}^3 \times \mathbb{P}^3 \hookrightarrow \mathbb{P}^{15}$. *The parameter space* Θ *is* 13*-dimensional, but the model* $\mathcal{M}^{(2)}$ *is* 11*-dimensional, so the given parametrization is not identifiable (Feinberg et al., 2007). This means that the Hessian matrix* $\mathbf{H}$ *is singular, and hence the Laplace approximation (29) is not defined.*

Example 8 *We compute (29) and (30) for the model and data in Example 3. According to Hoşten et al. (2005, Example 9), the likelihood function* $p_0^{51}p_1^{18}p_2^{73}p_3^{25}p_4^{75}$ *has three local maxima* $(\hat{p}_0,\hat{p}_1,\hat{p}_2,\hat{p}_3,\hat{p}_4)$ *in the model* $\mathcal{M}^{(2)}$*, and these translate into six local maxima* $(\hat{\sigma},\hat{\theta},\hat{\rho})$ *in the parameter space* Θ*, which is the 3-cube. The two global maxima are*

$$(0.3367691969, 0.0287713237, 0.6536073424),$$
$$(0.6632308031, 0.6536073424, 0.0287713237).$$

Both of these points in Θ *give the same point in the model:*

$$(\hat{p}_0,\hat{p}_1,\hat{p}_2,\hat{p}_3,\hat{p}_4) \;=\; (0.12104, 0.25662, 0.20556, 0.10758, 0.30920).$$

The likelihood function evaluates to $0.1395471101 \times 10^{-18}$ *at this point. The following table compares the various approximations. Here, "Actual" refers to the base-10 logarithm of the marginal likelihood in Example 3.*

BIC	*-22.43100220*
Laplace	*-22.39666281*
Actual	*-22.10853411*

The method for computing the marginal likelihood which was found to be most accurate in the experimental study is the *candidate method* (Chickering and Heckerman, 1997, §3.4). This is a Monte-Carlo method which involves running a Gibbs sampler. The basic idea is that one wishes to compute a large sum, such as (20) by sampling among the terms rather than listing all terms. In the candidate method one uses not the sum (20) over the lattice points in the zonotope but the more naive sum over all 2^N hidden data that would result in the observed data represented by U. The value of the sum is the number of terms, 2^N, times the average of the summands, each of which is easy to compute. A comparison of the results of the candidate method with our exact computations, as well as a more accurate version of Gibbs sampling which is adapted for (20), will be the subject of a future study.

One of the applications of marginal likelihood integrals lies in model selection. An important concept in that field is that of *Bayes factors*. Given data and two competing models, the Bayes factor is the ratio of the marginal likelihood integral of the first model over the marginal likelihood integral of the second model. In our context it makes sense to form that ratio for the independence model $\mathcal{M}$ and its mixture $\mathcal{M}^{(2)}$. To be precise, given any independence model, specified by positive integers $s_1,\ldots,s_k,t_1,\ldots,t_k$ and a corresponding data vector $U \in \mathbb{N}^n$, the Bayes factor is the ratio of the marginal likelihood in Lemma 1 and the marginal likelihood in Theorem 4. Both quantities are rational numbers and hence so is their ratio.

Corollary 10 *The Bayes factor which discriminates between the independence model* $\mathcal{M}$ *and the mixture model* $\mathcal{M}^{(2)}$ *is a rational number. It can be computed exactly using Algorithm 2 (and our* MAPLE*-implementation).*

Example 9 *We conclude by applying our method to a data set taken from the Bayesian statistics literature. Evans, Gilula, and Guttman (1989, §3) analyzed the association between length of hospital stay (in years Y) of* 132 *schizophrenic patients and the frequency with which they are visited*

by relatives. Their data set is the following 3×3 *contingency table:*

			$2\le Y<10$	$10\le Y<20$	$20\le Y$	*Totals*
		Visited regularly	43	16	3	*62*
U	$=$	*Visited rarely*	6	11	10	*27*
		Visited never	9	18	16	*43*
		Totals	*58*	*45*	*29*	**132**

They present estimated posterior means and variances for these data, where "each estimate requires a 9-dimensional integration" *(Evans et al., 1989, p. 561). Computing their integrals is essentially equivalent to ours, for* $k=2, s_1=s_2=1, t_1=t_2=2$ *and* $N=132$. *The authors emphasize that* "the dimensionality of the integral does present a problem" *(Evans et al., 1989, p. 562), and they point out that* "all posterior moments can be calculated in closed form .... however, even for modest N these expressions are far to complicated to be useful" *(Evans et al., 1989, p. 559).*

We differ on that conclusion. In our view, the closed form expressions in Section 3 are quite useful for modest sample size N. *Using Algorithm 2, we computed the integral (25). It is the rational number with numerator*

$$\begin{aligned}&27801948853106338912064360032498932910387614080 5\\ &28524283958209256935726588667532284587409752803 3\\ &99493069713103633199906939405711180837568853737\end{aligned}$$

and denominator

$$\begin{aligned}&12288402873591935400678094796599848745442833177572204\\ &50448819979286456995185542195946815073112429169997801\\ &33503900169921912167352239204153786645029153951176422\\ &43298328046163472261962028461650432024356339706541132\\ &34375318471880274818667657423749120000000000000000.\end{aligned}$$

To obtain the marginal likelihood for the data U *above, that rational number (of moderate size) still needs to be multiplied with the normalizing constant*

$$\frac{132!}{43!\cdot 16!\cdot 3!\cdot 6!\cdot 11!\cdot 10!\cdot 9!\cdot 18!\cdot 16!}.$$

Acknowledgments

Shaowei Lin was supported by graduate fellowship from A*STAR (Agency for Science, Technology and Research, Singapore). Bernd Sturmfels was supported by an Alexander von Humboldt research prize and the U.S. National Science Foundation (DMS-0456960). Zhiqiang Xu was supported by the NSFC grant 10871196 and a Sofia Kovalevskaya prize awarded to Olga Holtz.

References

D.M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of bayesian networks with hidden variables. *Machine Learning*, 29:181–212, 1997. Microsoft Research Report, MSR-TR-96-08.

D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer-Verlag, 2 edition, 2005.

M. Drton. Likelihood ratio tests and singularities. *Ann. Statist.*, 37(2):979–1012, 2009.

M. Drton, B. Sturmfels, and S. Sullivant. *Lectures on Algebraic Statistics*, volume 39 of *Oberwolfach Seminars*. Birkhäuser, Basel, 2009.

M. Evans, Z. Gilula, and I. Guttman. Latent class analysis of two-way contingency tables by bayesian methods. *Biometrika*, 76:557–563, 1989.

S. Feinberg, P. Hersh, A. Rinaldo, and Y. Zhou. Maximum likelihood estimation in latent class models for contingency table data. arXiv:0709.3535, 2007.

W. Fulton. *Introduction to Toric Varieties*. Princeton Univ. Press, 1993.

S. Gao, G. Jiang, and M. Zhu. Solving the 100 swiss francs problem. arXiv:0809.4627, 2008.

D. Geiger and D. Rusakov. Asymptotic model selection for naive bayesian networks. *Journal of Machine Learning Research*, 6:1–35, 2005.

S. Hoşten, A. Khetan, and B. Sturmfels. Solving the likelihood equations. *Foundations of Computational Mathematics*, 5:389–407, 2005.

L. Pachter and B. Sturmfels. *Algebraic Statistics for Computational Biology*. Cambridge University Press, 2005.

R. Stanley. A zonotope associated with graphical degree sequences. In P. Gritzmann and B. Sturmfels, editors, *Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift*, volume 4 of *DIMACS Series in Discrete Mathematics*, pages 555–570. Amer. Math. Soc., 1991.

B. Sturmfels. Open problems in algebraic statistics. In M. Putinar and S. Sullivant, editors, *Emerging Applications of Algebraic Geometry*, volume 149 of *Volumes in Mathematics and its Applications*, pages 351–364. I.M.A., 2008.

S. Watanabe. Algebraic analysis for nonidentifiable learning machines. *Neural Computation*, 13: 899–933, 2001.

S. Watanabe and K. Yamazaki. Singularities in mixture models and upper bounds of stochastic complexity. *International Journal of Neural Networks*, 16:1029–1038, 2003.

S. Watanabe and K. Yamazaki. Newton diagram and stochastic complexity in mixture of binomial distributions. In *Algorithmic Learning Theorem*, volume 3244 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2004.

G. Ziegler. *Lectures on Polytopes*. Springer-Verlag, 1995.

Journal of Machine Learning Research 10 (2009) 1633-1685 Submitted 6/08; Revised 5/09; Published 7/09

Transfer Learning for Reinforcement Learning Domains: A Survey

Matthew E. Taylor* TAYLORM@USC.EDU
Computer Science Department
The University of Southern California
Los Angeles, CA 90089-0781

Peter Stone PSTONE@CS.UTEXAS.EDU
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

Editor: Sridhar Mahadevan

Abstract

The reinforcement learning paradigm is a popular way to address problems that have only limited environmental feedback, rather than correctly labeled examples, as is common in other machine learning contexts. While significant progress has been made to improve learning in a single task, the idea of *transfer learning* has only recently been applied to reinforcement learning tasks. The core idea of transfer is that experience gained in learning to perform one task can help improve learning performance in a related, but different, task. In this article we present a framework that classifies transfer learning methods in terms of their capabilities and goals, and then use it to survey the existing literature, as well as to suggest future directions for transfer learning work.

Keywords: transfer learning, reinforcement learning, multi-task learning

1. Transfer Learning Objectives

In *reinforcement learning* (RL) (Sutton and Barto, 1998) problems, leaning agents take sequential actions with the goal of maximizing a reward signal, which may be time-delayed. For example, an agent could learn to play a game by being told whether it wins or loses, but is never given the "correct" action at any given point in time. The RL framework has gained popularity as learning methods have been developed that are capable of handling increasingly complex problems. However, when RL agents begin learning *tabula rasa*, mastering difficult tasks is often slow or infeasible, and thus a significant amount of current RL research focuses on improving the speed of learning by exploiting domain expertise with varying amounts of human-provided knowledge. Common approaches include deconstructing the task into a hierarchy of subtasks (cf., Dietterich, 2000); learning with higher-level, temporally abstract, actions (e.g., *options*, Sutton et al. 1999) rather than simple one-step actions; and efficiently abstracting over the state space (e.g., via function approximation) so that the agent may generalize its experience more efficiently.

The insight behind *transfer learning* (TL) is that generalization may occur not only within tasks, but also *across tasks*. This insight is not new; transfer has long been studied in the psychological literature (cf., Thorndike and Woodworth, 1901; Skinner, 1953). More relevant are a number of

∗. The first author wrote the majority of this article while a graduate student at the University of Texas at Austin.

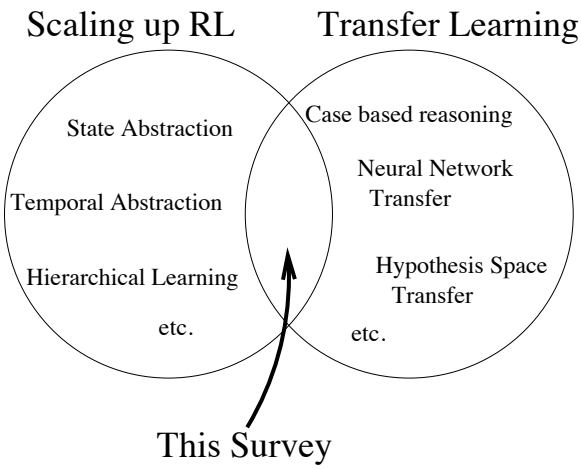


Figure 1: This article focuses on transfer between reinforcement learning tasks.

approaches that transfer between machine learning tasks (Caruana, 1995; Thrun, 1996), for planning tasks (Fern et al., 2004; Ilghami et al., 2005), and in the context of cognitive architectures (Laird et al., 1986; Choi et al., 2007). However, TL for RL tasks has only recently been gaining attention in the artificial intelligence community. Others have written surveys for reinforcement learning (Kaelbling et al., 1996), and for transfer across machine learning tasks (Thrun and Pratt, 1998), which we will not attempt to duplicate; this article instead focuses on transfer *between RL tasks* (see Figure 1) to provide an overview of a new, growing area of research.

Transfer learning in RL is an important topic to address at this time for three reasons. First, in recent years RL techniques have achieved notable successes in difficult tasks which other machine learning techniques are either unable or ill-equipped to address (e.g., TDGammon Tesauro 1994, job shop scheduling Zhang and Dietterich 1995, elevator control Crites and Barto 1996, helicopter control Ng et al. 2004, marble maze control Bentivegna et al. 2004, Robot Soccer Keepaway Stone et al. 2005, and quadruped locomotion Saggar et al. 2007 and Kolter et al. 2008). Second, classical machine learning techniques such as rule induction and classification are sufficiently mature that they may now easily be leveraged to assist with TL. Third, promising initial results show that not only are such transfer methods possible, but they can be very effective at speeding up learning. The 2005 DARPA Transfer Learning program (DARPA, 2005) helped increase interest in transfer learning. There have also been some recent workshops providing exposure for RL techniques that use transfer. The 2005 NIPS workshop, "Inductive Transfer: 10 Years Later," (Silver et al., 2005) had few RL-related transfer papers, the 2006 ICML workshop, "Structural Knowledge Transfer for Machine Learning," (Banerjee et al., 2006) had many, and the 2008 AAAI workshop, "Transfer Learning for Complex Tasks," (Taylor et al., 2008a) focused on RL.

1.1 Paper Overview

The goals of this survey are to introduce the reader to the transfer learning problem in RL domains, to organize and discuss current transfer methods, and to enumerate important open questions in RL transfer. In transfer, knowledge from one or more *source task(s)* is used to learn one or more *target task(s)* faster than if transfer was not used. The literature surveyed is structured primarily by grouping methods according to how they allow source and target tasks to differ. We further

distinguish methods according to five different dimensions (see Section 2.2). Some of the questions that distinguish transfer methods include:

- What are the goals of the transfer method? By what metric(s) will success be measured? Section 2 examines commonly used metrics, as well as different settings where transfer learning can improve learning.
- What assumptions, if any, are made regarding the similarity between the tasks? Section 3.2.1 enumerates common differences, such as changes to the space in which agents operate, allowing the agents to have different goals, or letting agents have different sets of actions.
- How does a transfer method identify what information can/should be transferable? Section 3.2.2 enumerates possibilities ranging from assuming *all* previously seen tasks are directly useful to autonomously learning which source task(s) are useful for learning in the current target task.
- What information is transferred between tasks? Section 3.2.3 discusses possibilities ranging from very low-level information (such as direct control knowledge) to high-level information (such as rules regarding how a particular domain functions).

The following section presents a discussion about how to best evaluate transfer in RL. There are many different situations in which transfer can be useful and these different situations may entail different metrics. This discussion will prepare the reader to better understand how transfer may be used. Section 3.1 will briefly discuss reinforcement learning and the notation used in the article. Section 3.2 enumerates the ways in which transfer methods can differ, providing a skeleton for the structure of this survey. Sections 3.3 and 3.4 provide additional high-level categorization of TL methods and Section 3.5 discusses related learning paradigms which are explicitly not discussed in this survey.

The bulk of the remainder of the article (Sections 4–8) discuss contemporary TL methods, arranged by the goals of, and methods employed by, the designers. Lastly, Section 9 discusses current open questions in transfer and concludes.

2. Evaluating Transfer Learning Methods

Transfer techniques assume varying degrees of autonomy and make many different assumptions. To be fully autonomous, an RL transfer agent would have to perform all of the following steps:

1. Given a target task, select an appropriate source task or set of tasks from which to transfer.
2. Learn how the source task(s) and target task are related.
3. Effectively transfer knowledge from the source task(s) to the target task.

While the mechanisms used for these steps will necessarily be interdependent, TL research has focused on each independently, and no TL methods are currently capable of robustly accomplishing all three goals.

A key challenge in TL research is to define evaluation metrics, precisely because there are many possible measurement options and algorithms may focus on any of the three steps above. This section focuses on how to best evaluate TL algorithms so that the reader may better understand the

different goals of transfer and the situations where transfer may be beneficial.[1] For instance, it is not always clear how to treat learning in the source task: whether to charge it to the TL algorithm or to consider it as a "sunk cost." On the one hand, a possible goal of transfer is to reduce the overall time required to learn a complex task. In this scenario, a *total time scenario*, which explicitly includes the time needed to learn the source task or tasks, would be most appropriate. On the other hand, a second reasonable goal of transfer is to effectively reuse past knowledge in a novel task. In this case, a *target task time scenario*, which only accounts for the time spent learning in the target task, is reasonable.

The total time scenario may be more appropriate when an agent is explicitly guided by a human. Suppose that a user wants an agent to learn to perform a task, but recognizes that the agent may be able to learn a sequence of tasks faster than if it directly tackled the difficult task. The human can construct a series of tasks for the agent, suggesting to the agent how the tasks are related. Thus the agent's TL method will easily accomplish steps 1 and 2 above, but it must efficiently transfer knowledge between tasks (step 3). To successfully transfer in this setting, the agent would have to learn the entire sequence of tasks faster than if it had spent its time learning the final target task directly (see the total time scenario in Figure 2).

The target task time scenario is more appropriate for a fully autonomous learner. A fully autonomous agent must be able to perform steps 1–3 on its own. However, metrics for this scenario do not need to take into account the cost of learning source tasks. The target task time scenario emphasizes the agent's ability to use knowledge from one or more previously learned source tasks without being charged for the time spent learning them (see the target task time scenario in Figure 2). In this survey we will see that the majority of existing transfer algorithms assume a human-guided scenario, but disregard time spent training in the source task. When discussing individual TL methods, we will specifically call attention to the methods that do account for the total training time and do not treat the time spent learning a source task as a sunk cost.

Many metrics to measure the benefits of transfer are possible (shown in Figure 3, replicated from our past transfer learning work, Taylor and Stone 2007b):

1. *Jumpstart*: The initial performance of an agent in a target task may be improved by transfer from a source task.

2. *Asymptotic Performance*: The final learned performance of an agent in the target task may be improved via transfer.

3. *Total Reward*: The total reward accumulated by an agent (i.e., the area under the learning curve) may be improved if it uses transfer, compared to learning without transfer.

4. *Transfer Ratio*: The ratio of the total reward accumulated by the transfer learner and the total reward accumulated by the non-transfer learner.

5. *Time to Threshold*: The learning time needed by the agent to achieve a pre-specified performance level may be reduced via knowledge transfer.

Metrics 1–4 are most appropriate in the fully autonomous scenario as they do not charge the agent for time spent learning any source tasks. To measure the total time, the metric must account for time

1. Evaluation is particularly important because there are very few theoretical results supporting TL for RL methods, as discussed further in Section 9.3. Instead, practitioners rely on empirical methods to evaluate the efficacy of their methods.

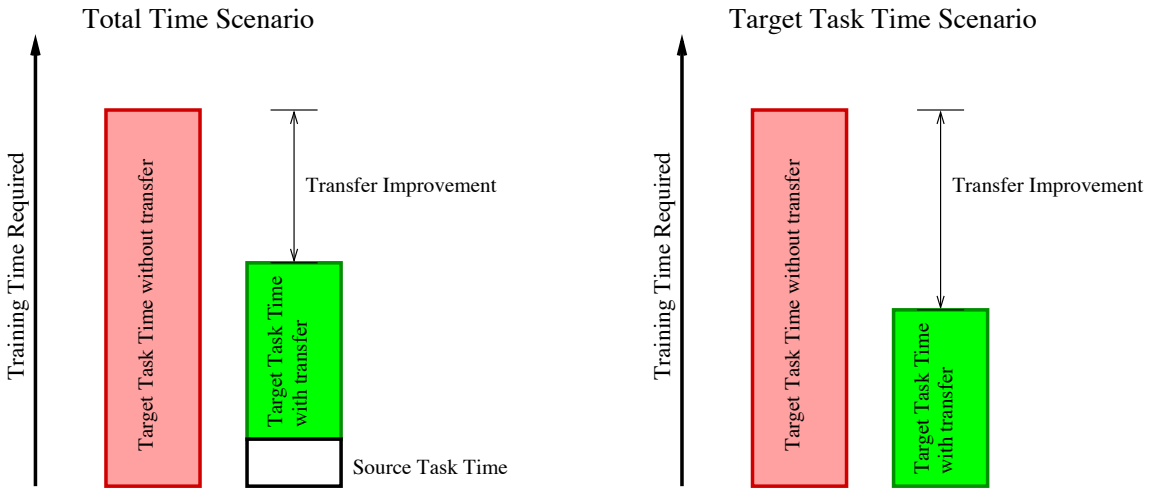


Figure 2: Successful TL methods may be able to reduce the total training time (left). In some scenarios, it is more appropriate to treat the source task time as a sunk cost and test whether the method can effectively reuse past knowledge to reduce the target task time (right).

spent learning one or more source tasks, which is natural when using metric 5. Other metrics have been proposed in the literature, but we choose to focus on these five because they are sufficient to describe the methods surveyed in this article.

For this article, we may think of learning time as a surrogate for *sample complexity*. Sample complexity (or data complexity) in RL refers to the amount of data required by an algorithm to learn. It is strongly correlated with learning time because RL agents only gain data by collecting it through repeated interactions with an environment.

2.1 Empirical Transfer Comparisons

The previous section enumerated five possible TL metrics, and while others are possible, these represent the methods most commonly used. However, each metric has drawbacks and none are sufficient to fully describe the benefits of any transfer method. Rather than attempting to create a total order ranking of different methods, which may indeed by impossible, we instead suggest that a multi-dimensional evaluation with multiple metrics is most useful. Specifically, some methods may "win" on a set of metrics relative to other methods, but "lose" on a different set. As the field better understands why different methods achieve different levels of success on different metrics, it should become easier to map TL methods appropriately to TL problems. Although the machine learning community has defined standard metrics (such as precision vs. recall curves for classification and mean squared error for regression), RL has no such standard. Empirically comparing two RL algorithms is a current topic of debate within the community, although there is some process towards standardizing comparisons (Whiteson et al., 2008). Theoretical comparisons are also not clear-cut, as samples to convergence, asymptotic performance, and the computational complexity are all valid axes along which to evaluate RL algorithms.

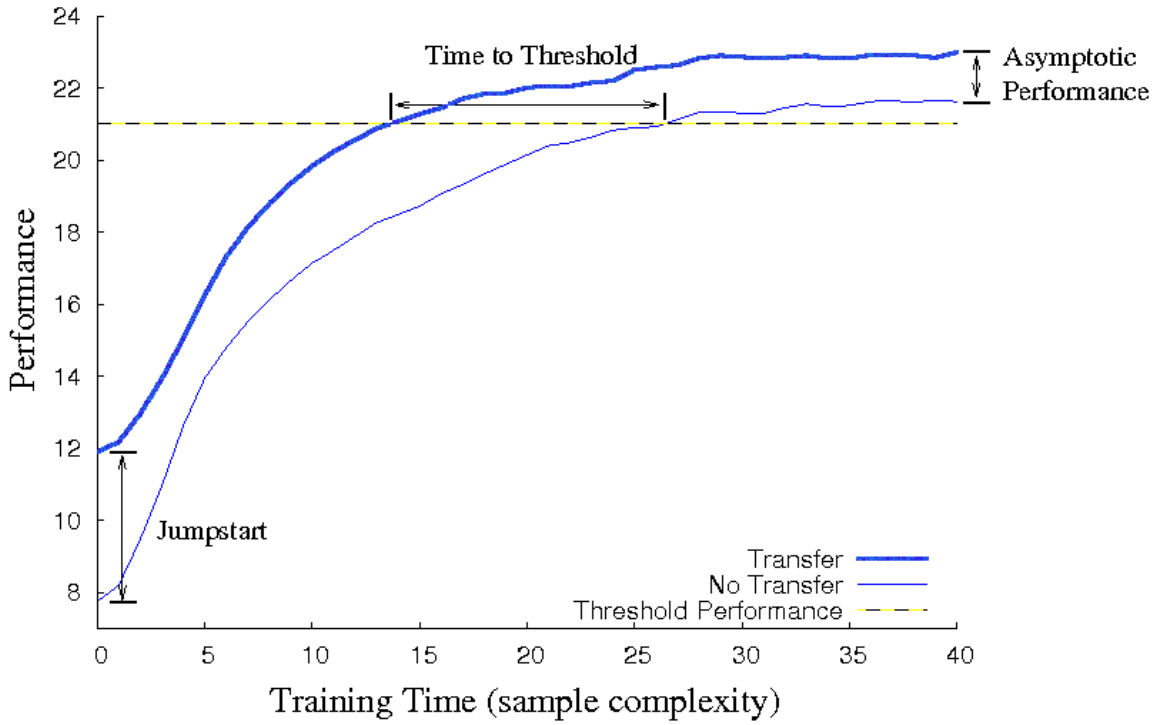


Figure 3: Many different metrics for measuring TL are possible. This graph show benefits to the jumpstart, asymptotic performance, time to threshold, and total reward (the area under the learning curve).

The first proposed transfer measure considers the agent's initial performance in a target task and answers the question, "can transfer be used so that the initial performance is increased relative to the performance of an initial (random) policy?" While such an initial jumpstart is appealing, such a metric fails to capture the behavior of *learning* in the target task and instead only focuses on the performance before learning occurs.

Asymptotic performance, the second proposed metric, compares the final performance of learners in the target task both with and without transfer. However, it may be difficult to tell when the learner has indeed converged (particularly in tasks with infinite state spaces) or convergence may take prohibitively long. In many settings the number of samples required to learn is most critical, not the performance of a learner with an infinite number of samples. Further, it is possible for different learning algorithms to converge to the same asymptotic performance but require very different numbers of samples to reach the same performance.

A third possible measure is that of the total reward accumulated during training. Improving initial performance and achieving a faster learning rate will help agents accumulate more on-line reward. RL methods are often not guaranteed to converge with function approximation and even when they do, learners may converge to different, sub-optimal performance levels. If enough samples are provided to agents (or, equivalently, learners are provided sufficient training time), a learning method which achieves a high performance relatively quickly will have less total reward than a learning method which learns very slowly but eventually plateaus at a slightly higher performance level. This metric is most appropriate for tasks that have a well-defined duration.

A fourth measure of transfer efficacy is that of the ratio of areas defined by two learning curves. Consider two learning curves in the target task where one uses transfer and one does not. Assuming that the transfer learner accrues more reward, the area under the transfer leaning curve will be greater than the area under the non-transfer learning curve. The ratio

$$r = \frac{\textit{area under curve with transfer - area under curve without transfer}}{\textit{area under curve without transfer}}$$

gives a metric that quantifies improvement from TL. This metric is most appropriate if the same final performance is achieved, or there is a predetermined time for the task. Otherwise the ratio will directly depend on how long the agents act in the target task.

While such a metric may be appealing as a candidate for inter-task comparisons, we note that the transfer ratio is not scale invariant. For instance, if the area under the transfer curve were 1000 units and the area under the non-transfer curve were 500, the transfer ratio would be 1.0. If all rewards were multiplied by a constant, this ratio would not change. But if an offset were added (e.g., each agent is given an extra +1 at the end of each episode, regardless of the final state), the ratio would change. The evaluation of a TL algorithm with the transfer ratio is therefore closely related to the reward structure of the target task being tested. Lastly, we note that although none of the papers surveyed in this article use such a metric, we hope that it will be used more often in the future.

The final metric, Time to Threshold, suffers from having to specify a (potentially arbitrary) performance agents must achieve. While there have been some suggestions how to pick such thresholds appropriately (Taylor et al., 2007a), the relative benefit of TL methods will clearly depend on the exact threshold chosen, which will necessarily be domain- and learning method-dependent. While choosing a range of thresholds to compare over may produce more representative measures (cf., Taylor et al., 2007b), this leads to having to generating a time vs. threshold curve rather than producing a single real valued number that evaluates a transfer algorithm's efficacy.

A further level of analysis that could be combined with any of the above methods would be to calculate a ratio comparing the performance of a TL algorithm with that of a human learner. For instance, a set of human subjects could learn a given target task with and without having first trained on a source task. By averaging over their performances, different human transfer metrics could be calculated and compared to that of a TL algorithm. However, there are many ways to manipulate such a meta-metric. For instance, if a target task is chosen that humans are relatively proficient at, transfer will provide them very little benefit. If that same target task is difficult for a machine learning algorithm, it will be relatively easy to show that the TL algorithm is quite effective relative to human transfer, even if the agent's absolute performance is extremely poor.

A major drawback of all the metrics discussed is that none are appropriate for inter-domain comparisons. The vast majority of papers in this survey compare learning with and without transfer—their authors often do not attempt to directly compare different transfer methods. Developing fair metrics that apply across multiple problem domains would facilitate better comparisons of methods. Such inter-domain metrics may be infeasible in practice, in which case standardizing on a set of test domains would assist in comparing different TL methods (as discussed further in Section 9). In the absence of either a set of inter-domain metrics or a standard benchmark suite of domains, we limit our comparisons of different TL methods in this survey to their applicability, assumptions, and algorithmic differences. When discussing different methods, we may opine on the method's relative

performance, but we remind the reader that such commentary is largely based on intuition rather than empirical data.

2.2 Dimensions of Comparison

In addition to differing on evaluation metrics, we categorize TL algorithms along five dimensions, which we use as the main organizing framework for our survey of the literature:

I *Task difference assumptions*: What assumptions does the TL method make about how the source and target are allowed to differ? Examples of things that can differ between the source and target tasks include different system dynamics (i.e., the target task becomes harder to solve is some incremental way), or different sets of possible actions at some states. Such assumptions define the types of source and target tasks that the method can transfer between. Allowing transfer to occur between less similar source and target tasks gives more flexibility to a human designer in the human-guided scenario. In the fully autonomous scenario, more flexible methods are more likely to be able to successfully apply past knowledge to novel target tasks.

II *Source task selection*: In the simplest case, the agent assumes that a human has performed source task selection (the human-guided scenario), and transfers from one or more selected tasks. More complex methods allow the agent to select a source task or set of source tasks. Such a selection mechanism may additionally be designed to guard against *negative transfer*, where transfer hurts the learner's performance. The more robust the selection mechanism, the more likely it is that transfer will be able to provide a benefit. While no definitive answer to this problem exists, successful techniques will likely have to account for specific target task characteristics. For instance, Carroll and Seppi (2005) motivate the need for general task similarity metrics to enable robust transfer, propose three different metrics, and then proceed to demonstrate that none is always "best," just as there is never a "best" inductive bias in a learning algorithm.

III *Task Mappings*: Many methods require a mapping to transfer effectively: in addition to knowing that a source task and target task are related, they need to know *how* they are related. *Inter-task mappings* (discussed in detail later in Section 3.4) are a way to define how two tasks are related. If a human is in the loop, the method may assume that such task mappings are provided; if the agent is expected to transfer autonomously, such mappings have to be learned. Different methods use a variety of techniques to enable transfer, both on-line (while learning the target task) and offline (after learning the source task but before learning the target task). Such learning methods attempt to minimize the number of samples needed and/or the computational complexity of the learning method, while still learning a mapping to enable effective transfer.

IV *Transferred Knowledge*: What type of information is transferred between the source and target tasks? This information can range from very low-level information about a specific task (i.e., the expected outcome when performing an action in a particular location) to general heuristics that attempt to guide learning. Different types of knowledge may transfer better or worse depending on task similarity. For instance, low-level information may transfer across closely related tasks, while high-level concepts may transfer across pairs of less similar tasks. The mechanism that transfers knowledge from one task to another is closely related to what

is being transferred, how the task mappings are defined (III), and what assumptions about the two tasks are made (I).

V *Allowed Learners*: Does the TL method place restrictions on what RL algorithm is used, such as applying only to temporal difference methods? Different learning algorithms have different biases. Ideally an experimenter or agent would select the RL algorithm to use based on characteristics of the task, not on the TL algorithm. Some TL methods require that the source and target tasks be learned with the same method, other allow a class of methods to be used in both tasks, but the most flexible methods decouple the agents' learning algorithms in the two tasks.

An alternate TL framework may be found in the related work section of Lazaric (2008), a recent PhD thesis on TL in RL tasks. Lazaric compares TL methods in terms of the type of benefit (jumpstart, total reward, and asymptotic performance), the allowed differences between source and target (different goal states, different transition functions but the same reward function, and different state and action spaces) and the type of transferred knowledge (experience or structural knowledge). Our article is more detailed both in the number of approaches considered, the depth of description about each approach, and also uses a different organizational structure. In particular, we specify which of the methods improve which of five TL metrics, we note which of the methods account for source task training time rather than treating it as a sunk cost, and we differentiate methods according to five dimensions above.

3. Transfer for Reinforcement Learning

In this section we first give a brief overview of notation. We then summarize the methods discussed in this survey using the five dimensions previously discussed, as well as enumerating the possible attributes for these dimensions. Lastly, learning paradigms with goals similar to transfer are discussed in Section 3.5.

3.1 Reinforcement Learning Background

RL problems are typically framed in terms of *Markov decision processes* (MDPs) (Puterman, 1994). For the purposes of this article, *MDP* and *task* are used interchangeably. In an MDP, there is some set of possible perceptions of the current *state* of the world, $s \in S$, and a learning agent has one or more initial starting states, $s_{initial}$. The *reward function*, $R : S \mapsto \mathbb{R}$, maps each state of the environment to a single number which is the instantaneous reward achieved for reaching the state. If the task is *episodic*, the agent begins at a start state and executes actions in the environment until it reaches a terminal state (one or more of the states in s_{final}, which may be referred to as a *goal state*), at which point the agent is returned to a start state. An agent in an episodic task typically attempts to maximize the average reward per episode. In non-episodic tasks, the agent attempts to maximize the total reward, which may be discounted. By using a discount factor, γ, the agent can weigh immediate rewards more heavily than future rewards, allowing it to maximize a non-infinite sum of rewards.

An agent knows its current state in the environment, $s \in S$.[2] TL methods are particularly relevant in MDPs that have a large or continuous state, as these are the problems which are slow to learn *tabula rasa* and for which transfer may provide substantial benefits. Such tasks typically factor the state using *state variables* (or *features*), so that $s = \langle x_1, x_2, \ldots, x_n \rangle$ (see Figure 4). The agent's observed state may be different from the true state if there is perceptual noise. The set A describes the *actions* available to the agent, although not every action may be possible in every state.[3] The *transition function*, $T : S \times A \mapsto S$, takes a state and an action and returns the state of the environment after the action is performed. Transitions may be non-deterministic, making the transition function a probability distribution function. A learner senses the current state, s, and typically knows A and what state variables comprise S; however, it is generally not given R or T.

A *policy*, $\pi : S \mapsto A$, fully defines how a learner interacts with the environment by mapping perceived environmental states to actions. The success of an agent is determined by how well it maximizes the total reward it receives in the long run while acting under some policy π. An *optimal policy*, π^*, is a policy which does maximize the expectation of this value. Any reasonable learning algorithm attempts to modify π over time so that the agent's performance approaches that of π^* in the limit.

There are many possible approaches to learning such a policy (depicted as a black box in Figure 4), including:

- *Temporal difference* (TD) methods, such as *Q-learning* (Sutton, 1988; Watkins, 1989) and *Sarsa* (Rummery and Niranjan, 1994; Singh and Sutton, 1996), learn by backing up experienced rewards through time. An estimated *action-value function*, $Q : S \times A \mapsto \mathbb{R}$ is learned, where $Q(s,a)$ is the expected return found when executing action a from state s, and greedily following the current policy thereafter. The current best policy is generated from Q by simply selecting the action that has the highest value for the current state. *Exploration*, when the agent chooses an action to learn more about the environment, must be balanced with *exploitation*, when the agent selects what it believes to be the best action. One simple approach that balances the two is ε-greedy action selection: the agent selects an random action with chance ε, and the current best action is selected with probability $1 - \varepsilon$ (where ε is in [0,1]).

- *Policy search* methods, such as policy iteration (dynamic programming), policy gradient (Williams, 1992; Baxter and Bartlett, 2001), and direct policy search (Ng and Jordan, 2000), are in some sense simpler than TD methods because they directly modify a policy over time to increase the expected long-term reward by using search or other optimization techniques.

- *Dynamic programming* (Bellman, 1957) approaches assume that a full model of the environment is known (i.e., S, A, T, and R are provided to the agent and are correct). No interaction with the environment is necessary, but the agent must iteratively compute approximations for the true value or action-value function, improving them over time.

- *Model-based* or *Model-learning* methods (Moore and Atkeson, 1993; Kearns and Singh, 1998) attempt to estimate the true model of the environment (i.e., T and R) by interacting

2. If the agent only receives *observations* and does not know the true state, the agent may treat approximate its true state as the observation (cf., Stone et al., 2005), or it may learn using the Partially Observable Markov Decision Process (POMDP) (cf., Kaelbling et al., 1998) problem formulation, which is beyond the scope of this survey.
3. Although possible in principle, we are aware of no TL methods currently address MDPs with continuous actions.

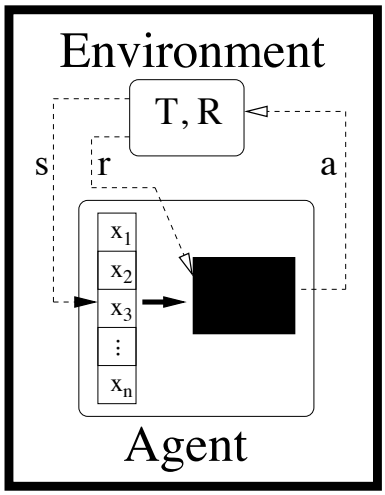


Figure 4: An agent interacts with an environment by sequentially selecting an action in an observed state, with the objective of maximizing an environmental reward signal.

with the environment over time. *Instance based methods* (Ormoneit and Sen, 2002) save observed interactions with the environment and leverage the instance directly to predict the model. *Bayesian RL* (Dearden et al., 1999) approaches use a mathematical model to explicitly represent uncertainty in the components of the model, updating expectations over time. The learned model is then typically used to help the agent decide how to efficiently explore or plan trajectories so that it can accrue higher rewards. While very successful in small tasks, few such methods handle continuous state spaces (cf., Jong and Stone, 2007), and they generally have trouble scaling to tasks with many state variables due to the "curse of dimensionality."

- *Relational reinforcement learning* (RRL) (Dzeroski et al., 2001) uses a different learning algorithm as well as a different state representation. RRL may be appropriate if the state of an MDP can be described in a relational or first-order language. Such methods work by reasoning over individual objects (e.g., a single block in a Blocksworld task) and thus may be robust to changes in numbers of objects in a task.

- *Batch* learning methods (e.g., *Least Squares Policy Iteration* (Lagoudakis and Parr, 2003) and Fitted-Q Iteration (Ernst et al., 2005) are offline and do not attempt to learn as the agent interacts with the environment. Batch methods are designed to be more sample efficient, as they can store a number of interactions with the environment and use the data multiple times for learning. Additionally, such methods allow a clear separation of the learning mechanism from the exploration mechanism (which much decide whether to attempt to gather more data about the environment or exploit the current best policy).

In tasks with small, discrete state spaces, Q and π can be fully represented in a table. As the state space grows, using a table becomes impractical, or impossible if the state space is continuous. In such cases, RL learning methods use *function approximators*, such as artificial neural networks, which rely on concise, parameterized functions and use supervised learning methods to set these parameters. Function approximation is used in large or continuous tasks to better generalize experience. Parameters and biases in the approximator are used to abstract the state space so that observed

data can influence a region of state space, rather than just a single state, and can substantially increase the speed of learning.

Some work in RL (Dean and Givan, 1997; Li et al., 2006; Mahadevan and Maggioni, 2007) has experimented with more systematic approaches to state abstractions (also called structural abstraction). Temporal abstractions have also been successfully used to increase the speed of learning. These macro-actions or *options* (Sutton et al., 1999) may allow the agent to leverage the sequence of actions to learn its task with less data. Lastly, hierarchical methods, such as MAXQ (Dietterich, 2000), allow learners exploit a task that is decomposed into different sub-tasks. The decomposition typically enables an agent to learn each subtask relatively quickly and then combine them, resulting in an overall learning speed improvement (compared to methods that do not leverage such a sub-task hierarchy).

3.2 Transfer Approaches

Having provided a brief overview of the RL notation used in this survey, we now enumerate possible approaches for transfer between RL tasks. This section lists attributes of methods used in the TL literature for each of the five dimensions discussed in Section 2.2, and summarizes the surveyed works in Table 1. The first two groups of methods apply to tasks which have the same state variables and actions. (Section 4 discusses the TL methods in the first block, and Section 5 discusses the multi-task methods in the second block.) Groups three and four consider methods that transfer between tasks with different state variables and actions. (Section 6 discusses methods that use a representation that does not change when the underlying MDP changes, while Section 7 presents methods that must explicitly account for such changes.) The last group of methods (discussed in Section 8) learns a mapping between tasks like those used by methods in the fourth group of methods. Table 2 concisely enumerates the possible values for the attributes, as well as providing a key to Table 1.

In this section the *mountain car* task (Moore, 1991; Singh and Sutton, 1996), a standard RL benchmark, will serve as a running example. In mountain car, an under-powered car moves along a curve and attempts to reach a goal state at the top of the right "mountain" by selecting between three actions on every timestep: {`Forward`, `Neutral`, `Backward`}, where `Forward` accelerates the car in the positive x direction and `Backward` accelerates the car in the negative x direction. The agent's state is described by two state variables: the horizontal position, x, and velocity, $\dot{x}$. The agent receives a reward of -1 on each time step. If the agent reaches the goal state the episode ends and the agent is reset to the start state (often the bottom of the hill, with zero velocity).

3.2.1 ALLOWED TASK DIFFERENCES

TL methods can transfer between MDPs that have different transition functions (denoted by t in Table 1), state spaces (s), start states (s_i), goal states (s_f), state variables (v), reward functions (r), and/or action sets (a). For two of the methods, the agent's representation of the world (the *agent-space*, describing physical sensors and actuators) remains the same, while the true state variables and actions (the *problem-space*, describing the task's state variables and macro-actions) can change (p in Table 1, discussed further in Section 6). There is also a branch of work that focuses on transfer between tasks which are composed of some number of objects that may change between the source and the target task, such as when learning with RRL (# in Table 1). When summarizing the allowed task differences, we will concentrate on the most salient features. For instance, when the source task

and target task are allowed to have different state variables and actions, the state space of the two tasks is different because the states are described differently, and the transition function and reward function must also change, but we only indicate "a" and "v."

These differences in the example mountain car task could be exhibited as:

- t: using a more powerful car motor or changing the surface friction of the hill
- s: changing the range of the state variables
- s_i: changing where the car starts each episode
- s_f: changing the goal state of the car
- v: describing the agent's state only by its velocity
- r: rather than a reward of -1 on every step, the reward could be a function of the distance from the goal state
- a: disabling the `Neutral` action
- p: the agent could describe the state by using extra state variables, such as the velocity on the previous timestep, but the agent only directly measures its current position and velocity
- #: the agent may need to control two cars simultaneously on the hill

3.2.2 SOURCE TASK SELECTION

The simplest method for selecting a source task for a given target task is to assume that only a single source task has been learned and that a human has picked it, assuring that the agent should use it for transfer (h in Table 1). Some TL algorithms allow the agent to learn multiple source tasks and then use them all for transfer (all). More sophisticated algorithms build a library of seen tasks and use only the most relevant for transfer (lib). Some methods are able to automatically modify a single source task so that the knowledge it gains from the modified task will likely be more useful in the target task (mod). However, none of the existing TL algorithms for RL can guarantee that the source tasks will be useful; a current open question is how to robustly avoid attempting to transfer from an irrelevant task.

3.2.3 TRANSFERRED KNOWLEDGE

The type of knowledge transferred can be primarily characterized by its specificity. Low-level knowledge, such as $\langle$ s, a, r, s$'$ $\rangle$ instances (I in Table 1), an action-value function (Q), a policy (π), a full task model (model), or prior distributions (pri), could all be directly leveraged by the TL algorithm to initialize a learner in the target task. Higher level knowledge, such as what action to use in some situations (A: a subset of the full set of actions), partial policies or options (π_p), rules or advice (rule), important features for learning (fea), proto-value functions (pvf: a type of learned feature), shaping rewards (R), or subtask definitions (sub) may not be directly used by the algorithm to fully define an initial policy, but such information may help guide the agent during learning in the target task.

3.2.4 TASK MAPPINGS

The majority of TL algorithms in this survey assume that no explicit task mappings are necessary because the source and target task have the same state variables and actions. In addition to having the same labels, the state variables and actions need to have the same semantic meanings in both tasks. For instance, consider again the mountain car domain. Suppose that the source task had the actions $A = \{$`Forward, Neutral, Backward`$\}$. If the target task had the actions $A = \{$`Right, Neutral, Left`$\}$, a TL method would need some kind of mapping because the actions had different labels. Furthermore, suppose that the target task had the same actions as the source ($A = \{$`Forward, Neutral, Backward`$\}$) but the car was facing the opposite direction, so that `Forward` accelerated the car in the negative `x` direction and `Backward` accelerated the car in the positive `x` direction. If the source and target task actions have different semantic meanings, there will also need to be some kind of inter-task mapping to enable transfer.

Methods that do not use a task mapping are marked as "N/A" in Table 1. TL methods which aim to transfer between tasks with different state variables or actions typically rely on a task mapping to define how the tasks are related (as defined in Section 3.4). Methods that use mappings and assume that they are human-supplied mappings are marked as "sup" in Table 1. A few algorithms leverage experience gained in the source task and target task (exp) or a high-level description of the MDPs in order to learn task mappings.

Methods using description-level knowledge differ primarily in what assumptions they make about what will be provided. One method assumes a qualitative understanding of the transition function (T), which would correspond to knowledge like "taking the action Neutral tends to have a positive influence on the velocity in the positive x direction." Two methods assume knowledge of one mapping (M_a: the "action mapping") to learn a second mapping (the "state variable mapping" in Section 3.4). Three methods assume that the state variables are "grouped" together to describe objects (sv_g). An example of the state variable grouping can be demonstrated in a mountain car task with multiple cars: if the agent knew which position state variables referred to the same car as certain velocity state variables, it would know something about the grouping of state variables. These different assumptions are discussed in detail in Section 8.

3.2.5 ALLOWED LEARNERS

The type of knowledge transferred directly affects the type of learner that is applicable (as discussed in Section 3.1). For instance, a TL method that transfers an action-value function would likely require that the target task agent use a temporal difference method to exploit the transferred knowledge. The majority of methods in the literature use a standard form of temporal difference learning (TD in Table 1), such as Sarsa. Other methods include Bayesian learning (B), hierarchical approaches (H), model-based learning (MB), direct policy search (PS), and relational reinforcement learning (RRL). Some TL methods focus on batch learning (Batch), rather than on-line learning. Two methods use *case based reasoning* (CBR) (Aamodt and Plaza, 1994) to help match previously learned instances with new instances, and one uses linear programming (LP) to calculate a value function from a given model (as part of a dynamic programming routine).

3.3 Multi-Task Learning

Closely related to TL algorithms, and discussed in Section 5, are *multi-task learning* (MTL) algorithms. The primary distinction between MTL and TL is that multi-task learning methods assume

Citation	Allowed Task Differences	Source Task Selection	Task Mappings	Transferred Knowledge	Allowed Learners	TL Metrics
Same state variables and actions: Section 4						
Selfridge et al. (1985)	t	h	N/A	Q	TD	tt†
Asada et al. (1994)	s_i	h	N/A	Q	TD	tt
Singh (1992)	r	all	N/A	Q	TD	ap, tr
Atkeson and Santamaria (1997)	r	all	N/A	model	MB	ap, j, tr
Asadi and Huber (2007)	r	h	N/A	π_p	H	tt
Andre and Russell (2002)	r, s	h	N/A	π_p	H	tr
Ravindran and Barto (2003b)	s, t	h	N/A	π_p	TD	tr
Ferguson and Mahadevan (2006)	r, s	h	N/A	pvf	Batch	tt
Sherstov and Stone (2005)	s_f, t	mod	N/A	A	TD	tr
Madden and Howley (2004)	s, t	all	N/A	rule	TD	tt, tr
Lazaric (2008)	s, t	lib	N/A	I	Batch	j, tr
Multi-Task learning: Section 5						
Mehta et al. (2008)	r	lib	N/A	π_p	H	tr
Perkins and Precup (1999)	t	all	N/A	π_p	TD	tt
Foster and Dayan (2004)	s_f	all	N/A	sub	TD, H	j, tr
Fernandez and Veloso (2006)	s_i, s_f	lib	N/A	π	TD	tr
Tanaka and Yamamura (2003)	t	all	N/A	Q	TD	j, tr
Sunmola and Wyatt (2006)	t	all	N/A	pri	B	j, tr
Wilson et al. (2007)	r, s_f	all	N/A	pri	B	j, tr
Walsh et al. (2006)	r, s	all	N/A	fea	any	tt
Lazaric (2008)*	r	all	N/A	fea	Batch	ap, tr
Different state variables and actions – no explicit task mappings: Section 6						
Konidaris and Barto (2006)	p	h	N/A	R	TD	j, tr
Konidaris and Barto (2007)	p	h	N/A	π_p	TD	j, tr
Banerjee and Stone (2007)	a, v	h	N/A	fea	TD	ap, j, tr
Guestrin et al. (2003)	#	h	N/A	Q	LP	j
Croonenborghs et al. (2007)	#	h	N/A	π_p	RRL	ap, j, tr
Ramon et al. (2007)	#	h	N/A	Q	RRL	ap, j, tt†, tr
Sharma et al. (2007)	#	h	N/A	Q	TD, CBR	j, tr
Different state variables and actions – inter-task mappings used: Section 7						
Taylor et al. (2007a)	a, v	h	sup	Q	TD	tt†
Taylor et al. (2007b)	a, v	h	sup	π	PS	tt†
Taylor et al. (2008b)	a, v	h	sup	I	MB	ap, tr
Torrey et al. (2005) Torrey et al. (2006)	a, r, v	h	sup	rule	TD	j, tr
Torrey et al. (2007)	a, r, v	h	sup	π_p	TD	j, tr
Taylor and Stone (2007b)	a, r, v	h	sup	rule	any/TD	j, tt†, tr
Learning inter-task mappings: Section 8						
Kuhlmann and Stone (2007)	a, v	h	T	Q	TD	j, tr
Liu and Stone (2006)	a, v	h	T	N/A	all	N/A
Soni and Singh (2006)	a, v	h	M_a, sv_g, exp	N/A	all	ap, j, tr
Talvitie and Singh (2007)	a, v	h	M_a, sv_g, exp	N/A	all	j
Taylor et al. (2007b)*	a, v	h	sv_g, exp	N/A	all	tt†
Taylor et al. (2008c)	a, v	h	exp	N/A	all	j, tr

Table 1: This table lists all the TL methods discussed in this survey and classifies each in terms of the five transfer dimensions (the key for abbreviations is in Table 2). Two entries, marked with a $\star$, are repeated due to multiple contributions. Metrics that account for source task learning time, rather than ignoring it, are marked with a †.

Allowed Task Differences	
a	action set may differ
p	problem-space may differ (agent-space must be identical)
r	reward function may differ
s_i	the start state may change
s_f	goal state may move
t	transition function may differ
v	state variables may differ
#	number of objects in state may differ

Source Task Selection	
all	all previously seen tasks are used
h	one source task is used (human selected)
lib	tasks are organized into a library and one or more may be used
mod	a human provides a source task that the agent automatically modifies

Task Mappings	
exp	agent learns the mappings from experience
M_a	the method must be provided with an action mapping (learns state variable mapping)
N/A	no mapping is used
sup	a human supplies the task mappings
sv_g	method is provided groupings of state variables
T	higher-level knowledge is provided about transfer functions to learn mapping

Transferred Knowledge	
A	an action set
fea	task features
I	experience instances
model	task model
π	policies
π_p	partial policies (e.g., options)
pri	distribution priors
pvf	proto-value function
Q	action-value function
R	shaping reward
rule	rules or advice
sub	subtask definitions

Allowed Learners	
B	Bayesian learner
Batch	batch learner
CBR	case based reasoning
H	hierarchical value-function learner
LP	linear programming
MB	model based learner
PS	policy search learner
RRL	relational reinforcement learning
TD	temporal difference learner

TL Metrics	
ap	asymptotic performance increased
j	jumpstart demonstrated
tr	total reward increased
tt	task learning time reduced

Table 2: This key provides a reference to the abbreviations in Table 1.

all problems experienced by the agent are drawn from the same distribution, while TL methods may allow for arbitrary source and target tasks. For example, a MTL task could be to learn a series of mountain car tasks, each of which had a transition function that was drawn from a fixed distribution of functions that specified a range of surface frictions. Because of this assumption, MTL methods generally do not need task mappings (dimension III in Section 2.2). MTL algorithms may be used to transfer knowledge between learners, similar to TL algorithms, or they can attempt to learn how to act on the entire class of problems.

When discussing supervised multitask learning (cf., Caruana, 1995, 1997), data from multiple tasks can be considered simultaneously. In an RL setting, rather than trying to learn multiple problems simultaneously (i.e., acting in multiple MDPs), agents tackle a sequence of tasks which are more closely related than in TL settings. It is possible that RL agents could learn multiple tasks simultaneously in a multiagent setting (Stone and Veloso, 2000), but this has not yet been explored in the literature. For the purposes of this survey, we will assume, as in other transfer settings, that tasks are learned in a sequential order.

Sutton et al. (2007) motivate this approach to transfer by suggesting that a single large task may be most appropriately tackled as a sequential series of subtasks. If the learner can track which subtask it is currently in, it may be able to transfer knowledge between the different subtasks, which are all presumably related because they are part of the same overall task. Such a setting may provide a well-grounded way of selecting a distribution of tasks to train over, either in the context of transfer or for multi-task learning. Note also that the additional assumptions in an MTL setting may be leveraged to allow a more rigorous theoretical analysis than in TL (cf., Kalmár and Szepesvári, 1999).

3.4 Inter-Task Mappings

Transfer methods that assume the source and target tasks use the same state variables and actions, as is the case in MTL, typically do not need an explicit mapping between task. In order to enable TL methods to transfer between tasks that do have such differences, the agent must know how the tasks are related. This section provides a brief overview of *inter-task mappings* (Taylor et al., 2007a), one formulation of task mappings. Task mappings like these are used by transfer methods discussed in Section 7.

To transfer effectively, when an agent is presented with a target task that has a set of actions (A'), it must know how those actions are related to the action set in the source task (A). (For the sake of exposition we focus on actions, but an analogous argument holds for state variables.) If the TL method knows that the two action sets are identical, no action mapping is necessary. However, if this is not the case, the agent needs to be told, or learn, how the two tasks are related. For instance, if the agent learns to act in a source task with the actions `Forward` and `Backward`, but the target task uses the actions `Right` and `Left`, the correspondence between these action sets may not be obvious. Even if the action labels were the same, if the actions had different semantic meanings, the default correspondence may be incorrect. Furthermore, if the cardinality of A and A' are not equal, there are actions without exact equivalences.

One option is to define an *action mapping* (χ_A) such that actions in the two tasks are mapped so that their effects are "similar," where similarity depends on the transfer and reward functions in the two MDPs.[4] Figure 5 depicts an action mapping as well as a *state-variable mapping* (χ_X) between two tasks. A second option is to define a *partial mapping* (Taylor et al., 2007b), such that any novel actions in the target task are ignored. Consider adding an action in a mountain car target task, `pull hand brake`, which did not have an analog in the source task. The partial mapping could map `Forward` to `Forward`, and `Backward` to `Backward`, but not map `pull hand brake` to any source task action. Because inter-task mappings are not functions, they are typically assumed to be easily invertible (i.e., mapping source task actions into target task actions, rather than target task actions to source task actions).

It is possible that mappings between states, rather than between state variables, could be used for transfer, although no work has currently explored this formulation.[5] Another possible extension is to link the mappings rather than making them independent. For instance, the action mapping could depend on the state that the agent is in, or the state variable mapping could depend on the action

4. An inter-task mapping often maps multiple entities in the target task to single entities in the source task because the target task is more complex than the source, but the mappings may be one-to-many, one-to-one, or many-to-many.
5. However, there are many possibilities for using this approach for transfer learning, such as through bisimulation (see Section 9).

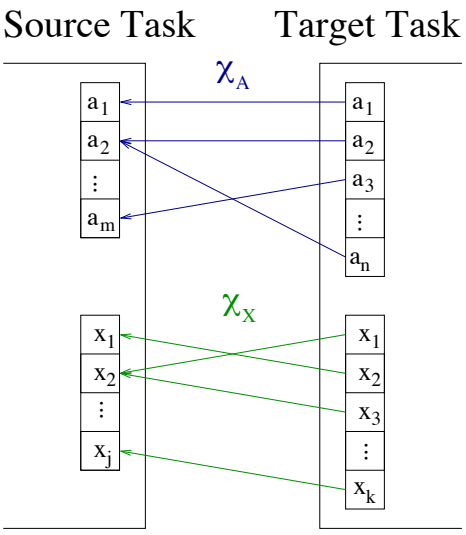


Figure 5: χ_A and χ_X are independent mappings that describe similarities between two MDPs. These mappings describe how actions in the target task are similar to actions in the source task and how state variables in the target task are similar to state variables in the source task, respectively.

selected. Though these extensions may be necessary based on the demands of particular MDPs, current methods have functioned well in a variety of tasks without such enhancements.

For a given pair of tasks, there could be many ways to formulate inter-task mappings. Much of the current TL work assumes that a human has provided a (correct) mapping to the learner. Work that attempts to learn a mapping that can be effectively used for transfer is discussed in Section 8.

3.5 Related Paradigms

In this survey, we consider transfer learning algorithms that use one or more source tasks to better learn in a different, but related, target task. There is a wide range of methods designed to improve the learning speed of RL methods. This section discusses four alternate classes of techniques for speeding up learning and differentiates them from transfer. While some TL algorithms may reasonably fit into one or more of the following categories, we believe that enumerating the types of methods *not* surveyed in this article will help clarify our subject of interest.

3.5.1 LIFELONG LEARNING

Thrun (1996) suggested the notion of lifelong learning where an agent may experience a sequence of tasks. Others (cf., Sutton et al., 2007) later extended this idea to the RL setting, suggesting than an agent interacting with the world for an extended period of time will necessarily have to perform in a sequence of tasks. Alternately, the agent may discover a series of spatially, rather than temporally, separated sub-tasks. Transfer would be a key component of any such system, but the lifelong learning framework is more demanding than that of transfer. First, transfer algorithms may reasonably focus on transfer between a single pair of related tasks, rather than attempting to account for any future task that an agent could encounter. Second, transfer algorithms are typically

told when a new task has begun, whereas in lifelong learning, agents may be reasonably expected to automatically identify new sub-tasks within the global MDP (i.e., the real world).

3.5.2 IMITATION LEARNING

The primary motivations for imitation methods are to allow agents to learn by watching another agent with similar abilities (Price and Boutilier, 2003; Syed and Schapier, 2007) or a human (Abbeel and Ng, 2005; Kolter et al., 2008) perform a task. Such algorithms attempt to learn a policy by observing an outside actor, potentially improving upon the inferred policy. In contrast, our definition of transfer learning focuses on agents successfully reusing internal knowledge on novel problems.

3.5.3 HUMAN ADVICE

There is a growing body of work integrating human advice into RL learners. For instance, a human may provide action suggestions to the agent (cf., Maclin and Shavlik, 1996; Maclin et al., 2005) or guide the agent through on-line feedback (cf., Knox and Stone, 2008). Leveraging humans' background and task-specific knowledge can significantly improve agents' learning ability, but it relies on a human being tightly integrated into the learning loop, providing feedback in an on-line manner. This survey instead concentrates on transfer methods in which a human is not continuously available and agents must learn autonomously.

3.5.4 SHAPING

Reward shaping (Colombetti and Dorigo, 1993; Mataric, 1994) in an RL context typically refers to allowing agent to train on an artificial reward signal rather than R. For instance, in the mountain car task, the agent could be given a higher reward as it gets closer to the goal state, rather then receiving -1 at every state except the goal. However, if the human can compute such a reward, s/he would probably already know the goal location, knowledge that the agent typically does not have. Additionally, the constructed reward function must be a potential function. If it is not, the optimal policy for the new MDP could be different from that of the original (Ng et al., 1999). A second definition of shaping follows Skinner's research (Skinner, 1953) where the reward function is modified over time in order to direct the behavior of the learner. This method, as well as the approach of using a static artificial reward, are ways of injecting human knowledge into the task definition to improve learning efficacy.

Erez and Smart (2008) have argued for a third definition of shaping as any supervised, iterative, process to assist learning. This includes modifying the dynamics of the task over time, modifying the internal learning parameters over time, increasing the actions available to the agent, and extending the agent's policy time horizon (e.g., as done in value iteration). All of these methods rely on a human to intelligently assist the agent in its learning task and may leverage transfer-like methods to successfully reuse knowledge between slightly different tasks. When discussing transfer, we will emphasize how knowledge is successfully reused rather than how a human may modify tasks to achieve the desired agent behavior improve agent learning performance.

3.5.5 REPRESENTATION TRANSFER

Transfer learning problems are typically framed as leveraging knowledge learned on a source task to improve learning on a related, but different, target task. Taylor and Stone (2007a) examine the

Citation	Allowed Task Differences	Source Task Selection	Task Mappings	Transferred Knowledge	Allowed Learners	TL Metrics
Same state variables and actions: Section 4						
Selfridge et al. (1985)	t	h	N/A	Q	TD	tt[†]
Asada et al. (1994)	s_i	h	N/A	Q	TD	tt
Singh (1992)	r	all	N/A	Q	TD	ap, tr
Atkeson and Santamaria (1997)	r	all	N/A	model	MB	ap, j, tr
Asadi and Huber (2007)	r	h	N/A	π_p	H	tt
Andre and Russell (2002)	r, s	h	N/A	π_p	H	tr
Ravindran and Barto (2003b)	s, t	h	N/A	π_p	TD	tr
Ferguson and Mahadevan (2006)	r, s	h	N/A	pvf	Batch	tt
Sherstov and Stone (2005)	s_f, t	mod	N/A	A	TD	tr
Madden and Howley (2004)	s, t	all	N/A	rule	TD	tt, tr
Lazaric (2008)	s, t	lib	N/A	I	Batch	j, tr

Table 3: This table reproduces the first group of methods from Table 1.

complimentary task of transferring knowledge between agents with different internal *representations* (i.e., the function approximator or learning algorithm) of the *same* task. Allowing for such shifts in representation gives additional flexibility to an agent designer; past experience may be transferred rather than discarded if a new representation is desired. A more important benefit is that changing representations partway through learning can allow agents to achieve better performance in less time. Selecting a representation is often key for solving a problem (cf., the *mutilated checkerboard problem* McCarthy 1964 where humans' internal representations of a problem drastically changes the problem's solvability) and different representations may make transfer more or less difficult. However, representation selection is a difficult problem in RL in general and discussions of representation selection (or its applications to transfer efficacy) are beyond the scope of this article.

4. Transfer Methods for Fixed State Variables and Actions

To begin our survey of TL methods, we examine the first group of methods in Table 1, reproduced in Table 3. These techniques may be used for transfer when the source and target tasks use the same state variables and when agents in both tasks have the same set of actions (see Figure 6).

In one of the earliest TL works for RL, Selfridge et al. (1985) demonstrated that it was faster to learn to balance a pole on a cart by changing the task's transition function, T, over time. The learner was first trained on a long and light pole. Once it successfully learned to balance the pole the task was made harder: the pole was shortened and made heavier. The total time spent training on a sequence of tasks and reusing the learned function approximator was faster than training on the hardest task directly.[6]

Similarly, the idea of *learning from easy missions* (Asada et al., 1994) also relies on a human constructing a set of tasks for the learner. In this work, the task (for example, a maze) is made incrementally harder not by changing the dynamics of the task, but by moving the agent's initial

6. As discussed in Section 3.5.3, we classify this work as transfer rather than as a "human advice" method; while the human may assist the agent in task selection, s/he does not provide direct on-line feedback while the agent learns.

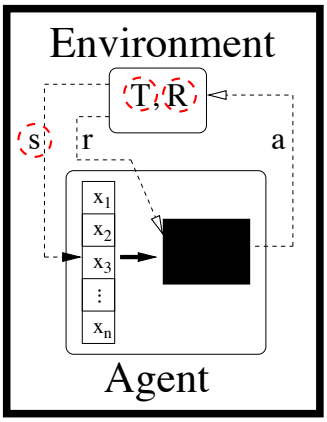


Figure 6: Methods in Section 4 are able to transfer between tasks that have different state spaces, different transition functions, and different reward functions, but only if the source and target tasks have the same actions and state variables. Dashed circles indicate the MDP components which may differ between the source task and target task.

state, $s_{initial}$, further and further from the goal state. The agent incrementally learns how to navigate to the exit faster than if it had tried to learn how to navigate the full maze directly. This method relies on having a known goal state from which a human can construct a series of source tasks of increasing difficulty.

Selfridge et al. (1985) and Asada et al. (1994) provide useful methods for improving learning, which follow from Skinner's animal training work. While they require a human to be in the loop, and to understand the task well enough to provide the appropriate guidance to the learner, these methods are relatively easy ways to leverage human knowledge. Additionally, they may be combined with many of the transfer methods that follow.

Rather than change a task over time, one could consider breaking down a task into a series of smaller tasks. This approach can be considered a type of transfer in that a single large target task can be treated as a series of simpler source tasks. Singh (1992) uses a technique he labels *compositional learning* to discover how to separate temporally sequential subtasks in a monolithic task. Each subtask has distinct beginning and termination conditions, and each subtask will be significantly easier to learn in isolation than in the context of the full task. Only the reward function, R, is allowed to change between the different subtasks and none of the other MDP components may vary, but the total reward can be increased. If subtasks in a problem are recognizable by state features, such subtasks may be automatically identified via vision algorithms (Drummond, 2002). Again, breaking a task into smaller subtasks can improve both the total reward and the asymptotic performance. This particular method is only directly applicable to tasks in which features clearly define subtasks due to limitations in the vision algorithm used. For instance, in a 2D navigation task each room may be a subtask and the steep value function gradient between impassable walls is easily identifiable. However, if the value function gradient is not distinct between different subtasks, or the subtask regions of state space are not polygonal, the algorithm will likely fail to automatically identify subtasks.

In Atkeson and Santamaria (1997), transfer between tasks in which only the reward function can differ are again considered. Their method successfully transfers a locally weighted regression

model of the transition function, which is learned in a source task, by directly applying it to a target task. Because their model enables planning over the transition function and does not account for the reward function, they show significant improvement to the jumpstart and total reward, as well as the asymptotic performance.

The next three methods transfer partial policies, or options, between different tasks. First, Asadi and Huber (2007) have the agent identify states that "locally form a significantly stronger 'attractor' for state space trajectories" as subgoals in the source task (i.e., a doorway between rooms that is visited relatively often compared to other parts of the state space). The agent then learns options to reach these subgoals via a learned action-value function, termed the *decision-level* model. A second action-value function, the *evaluation-level* model, includes all actions and the full state space. The agent selects actions by only considering the decision-level model but uses discrepancies between the two models to automatically increase the complexity of the decision-level model as needed. The model is represented as a *Hierarchical Bounded Parameter SMDP*, constructed so that the performance of an optimal policy in the simplified model will be within some fixed bound of the performance of the optimal policy on the initial model. Experiments show that transferring both the learned options and the decision-level representation allow the target task agent to learn faster on a task with a different reward function. In the roughly 20,000 target task states, only 81 distinct states are needed in the decision-level model, as most states do not need to be distinguished when selecting from learned options.

Second, Andre and Russell (2002) transfer learned subroutines between tasks, which are similar to options. The authors assume that the source and target tasks have a hierarchical structure, such as in the *taxi domain* (Dietterich, 2000). On-line analysis can uncover similarities between two tasks if there are only small differences in the state space (e.g., the state variables do not change) and then directly copy over the subroutine, which functions as a partial policy, thereby increasing the total reward in the target task. This method highlights the connection between state abstraction and transfer; if similarities can be found between parts of the state space in the two tasks, it is likely that good local controllers or local policies can be directly transferred.

Third, Ravindran and Barto (2003b) learn *relativized options* in a small, human selected source task. When learning in the target task, the agent is provided these options and a set of possible transformations it could apply to them so that they were relevant in the target task. For instance, if the source task were a small grid navigation task, the target task could be a large grid composed of rooms with similar shape to the source task and the transformations could be rotation and reflection operators. The agent uses experience in the target and Bayesian parameter estimation to select which transformations to use so that the target task's total reward is increased. Learning time in the source task is ignored, but is assumed to be small compared to the target task learning time.

Next, Ferguson and Mahadevan (2006) take a unique approach to transfer information about the source task's structure. *Proto-value functions* (PVFs) (Mahadevan and Maggioni, 2007) specify an ortho-normal set of basis functions, without regard to R, which can be used to learn an action-value function. After PVFs are learned in a small source task, they can be transferred to another discrete MDP that has a different goal or small changes to the state space. The target task can be learned faster and achieve higher total reward with the transferred PVFs than without. Additionally, the PVF can be scaled to larger tasks. For example, the target maze could have twice the width and height of the source maze: R, S, and T are all scaled by the same factor. In all cases only the target task time is counted.

The goal of learning PVFs is potentially very useful for RL in general and TL in particular. It makes intuitive sense that high-level information about how to best learn in a domain, such as appropriate features to reason over, may transfer well across tasks. There are few examples of meta-learners where TL algorithms learn high level knowledge to assist the agent in learning, rather than lower-level knowledge about how to act. However, we believe that there is ample room for such methods, including methods to learn other domain-specific learning parameters, such as learning rates, function approximator representations, an so on.

Instead of biasing the target task agent's learning representation by transferring a set of basis functions, Sherstov and Stone (2005) consider how to bias an agent by transferring an appropriate action set. If tasks have large action sets, all actions could be considered when learning each task, but learning would be much faster if only a subset of the actions needed to be evaluated. If a reduced action set is selected such that using it could produce near-optimal behavior, learning would be much faster with very little loss in final performance. The standard MDP formalism is modified so that the agent reasons about *outcomes* and *classes*. Informally, rather than reasoning over the probability of reaching a given state after an action, the learner reasons over the actions' effect, or outcome. States are grouped together in classes such that the probability of a given outcome from a given action will be the same for any state in a class. The authors then use their formalism to bound the value lost by using their abstraction of the MDP. If the source and target are very similar, the source task can be learned with the full action set, the optimal action set can be found from the learned Q-values, and learning the target with this smaller action set can speed up learning in the target task. The authors also introduce *random task perturbation* (RTP) which creates a *series* of source tasks from a single source task, thereby producing an action set which will perform well in target tasks that are less similar to the source task. Transfer with and without RTP is experimentally compared to learning without transfer. While direct action transfer can perform worse than learning without transfer, RTP was able to handle misleading source task experience so that performance was improved relative to no transfer in all target tasks and performance using the transferred actions approaches that of the optimal target task action set. Performance was judged by the total reward accumulated in the target task. Leffler et al. (2007) extends the work of Sherstov and Stone by applying the outcome/class framework to learn a *single* task significantly faster, and provides empirical evidence of correctness in both simulated and physical domains.

The idea of RTP is not only unique in this survey, but it is also potentially a very useful idea for transfer in general. While a number of TL methods are able to learn from a set of source tasks, no others attempt to automatically generate these source tasks. If the goal of an agent is perform as well as possible in a novel target task, it makes sense that the agent would try to train on many source tasks, even if they are artificial. How to best generate such source tasks so that they are most likely to be useful for an arbitrary target task in the same domain is an important area of open research.

Similar to previously discussed work (Selfridge et al., 1985; Asada et al., 1994), *Progressive RL* (Madden and Howley, 2004) is a method for transferring between a progression of tasks of increasing difficulty, but is limited to discrete MDPs. After learning a source task, the agent performs *introspection* where a symbolic learner extracts rules for acting based on learned Q-values from all previously learned tasks. The RL algorithm and introspection use different state features. Thus the two learning mechanisms learn in different state spaces, where the state features for the symbolic learner are higher-level and contain information otherwise hidden from the agent. When the agent acts in a novel task, the first time it reaches a novel state it initialize the Q-values of that state so that the action suggested by the learned rule is preferred. Progressive RL allows agents to learn infor-

mation in a set of tasks and then abstract the knowledge to a higher-level representation, allowing the agent to achieve higher total reward and reach the goal state for the first time faster. Time spent in the source task(s) is not counted.

Finally, Lazaric (2008) demonstrates that source task *instances* can be usefully transferred between tasks. After learning one or more source tasks, some experience is gathered in the target task, which may have a different state space or transition function. Saved instances (that is, observed $\langle s, a, r, s' \rangle$ tuples) are compared to instances from the target task. Instances from the source tasks that are most similar, as judged by their distance and alignment with target task data, are transferred. A batch learning algorithm then uses both source instances and target instances to achieve a higher reward and a jumpstart. *Region transfer* takes the idea one step further by looking at similarity with the target task per-sample, rather than per task. Thus, if source tasks have different regions of the state space which are more similar to the target, only those most similar regions can be transferred. In these experiments, time spent training in the target task is not counted towards the TL algorithm.

Region transfer is the only method surveyed which explicitly reasons about task similarity in *different parts* of the state space, and then selects source task(s) to transfer from. In domains where target tasks have regions of the state space that are similar to one or more source tasks, and other areas which are similar to other source tasks (or are similar to no source tasks), region transfer may provide significant performance improvements. As such, this method provides a unique approach to measuring, and exploiting, task similarity on-line. It is likely that this approach will inform future transfer methods, and is one possible way of accomplishing step # 1 in Section 2: Given a target task, select an appropriate source task from which to transfer, if one exists.

Taken together, these TL methods show that it is possible to efficiently transfer many different types of information between tasks with a variety of differences. It is worth re-emphasizing that many TL methods may be combined with other speedup methods, such as reward shaping, or with other transfer methods. For instance, when transferring between maze tasks, basis functions could be learned (Ferguson and Mahadevan, 2006) in the source task, a set of actions to transfer could be selected after training on a set of additional generated source tasks (Sherstov and Stone, 2005), and then parts of different source tasks could be leveraged to learn a target task (Lazaric, 2008). A second example would be to start with a simple source task and change it over time by modifying the transition function (Selfridge et al., 1985) and start state (Asada et al., 1994), while learning options (Ravindran and Barto, 2003b), until a difficult target task is learned. By examining how the source and target task differ and what base learning method is used, RL practitioners may select one or more TL method to apply to their domain of interest. However, in the absence of theoretical guarantees of transfer efficacy, any TL method has the potential to be harmful, as discussed further in Section 9.2.

5. Multi-Task Learning Methods

This section discusses scenarios where the source tasks and target task have the same state variables and actions. However, these methods (see Table 4, reproduced from Table 1) are explicitly MTL, and all methods in this section are designed to use multiple source tasks (see Figure 7). Some methods leverage all experienced source tasks when learning a novel target task and others are able to choose a subset of previously experienced tasks. Which approach is most appropriate depends on the assumptions about the task distribution: if tasks are expected to be similar enough that all past experience is useful, there is no need to select a subset. On the other hand, if the distribution of

Citation	Allowed Task Differences	Source Task Selection	Task Mappings	Transferred Knowledge	Allowed Learners	TL Metrics
Multi-Task learning: Section 5						
Mehta et al. (2008)	r	lib	N/A	π_p	H	tr
Perkins and Precup (1999)	t	all	N/A	π_p	TD	tt
Foster and Dayan (2004)	s_f	all	N/A	sub	TD, H	j, tr
Fernandez and Veloso (2006)	s_i, s_f	lib	N/A	π	TD	tr
Tanaka and Yamamura (2003)	t	all	N/A	Q	TD	j, tr
Sunmola and Wyatt (2006)	t	all	N/A	pri	B	j, tr
Wilson et al. (2007)	r, s_f	all	N/A	pri	B	j, tr
Walsh et al. (2006)	r, s	all	N/A	fea	any	tt
Lazaric (2008)	r	all	N/A	fea	Batch	ap, tr

Table 4: This table reproduces the group of MTL methods from Table 1.

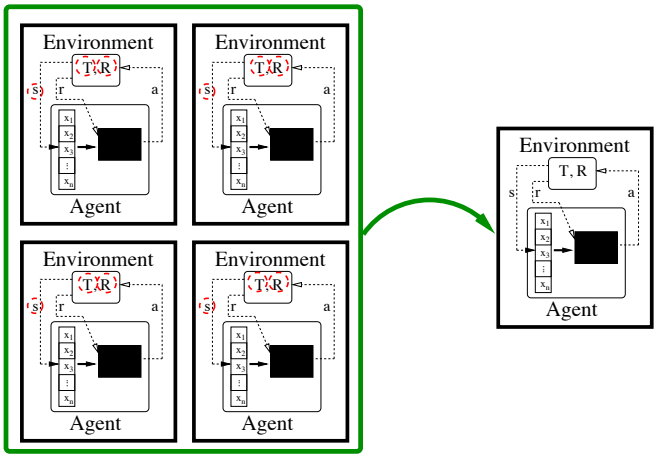


Figure 7: Multi-task learning methods assume tasks are chosen from a fixed distribution, use one or more source tasks to help learn the current task, and assume that all the tasks have the same actions and state variables. Dashed circles indicate the MDP components which may differ between tasks.

tasks is multi-modal, it is likely that transferring from all tasks is sub-optimal. None of the methods account for time spent learning in the source task(s) as the primary concern is effective learning on the next task chosen at random from an unknown (but fixed) distribution of MDPs.

Variable-reward hierarchical reinforcement learning (Mehta et al., 2008) assumes that the learner will train on a sequence of tasks which are identical except for different *reward weights*. The reward weights define how much reward is assigned via a linear combination of *reward features*. The authors provide the reward features to the agent for a given set of tasks. For instance, in a real-time strategy domain different tasks could change the reward features, such as the benefit from collecting units of gold or from damaging the enemy. However, it is unclear how many domains of interest have reward features, which are provided to the agent at the start of each task. Using a hierarchical RL method, subtask policies are learned. When a novel target task is encountered, the agent sets the initial policy to that of the most similar source task, as determined by the dot product with previ-

ously observed reward weight vectors. The agent then uses an ε-greedy action selection method at each level of the task hierarchy to decide whether to use the best known sub-task policy or explore. Some sub-tasks, such as navigation, will never need to be relearned for different tasks because they are unaffected by the reward weights, but any suboptimal sub-task policies will be improved. As the agent experiences more tasks, the total reward in each new target task increases, relative to learning the task without transfer.

A different problem formulation is posed by Perkins and Precup (1999) where the transition function, T, may change after reaching the goal. Upon reaching the goal, the agent is returned to the start state and is not told if, or how, the transition function has changed, but it knows that T is drawn randomly from some fixed distribution. The agent is provided a set of hand-coded options which assist in learning on this set of tasks. Over time, the agent learns an accurate action-value function over these options. Thus, a single action-value function is learned over a set of tasks, allowing the agent to more quickly reach the goal on tasks with novel transition functions.

Instead of transferring options, Foster and Dayan (2004) aim to identify sub-tasks in a source task and use this information in a target task, a motivation similar to that of Singh (1992). Tasks are allowed to differ in the placement of the goal state. As optimal value functions are learned in source tasks, an *expectation-maximization* algorithm (Dempster et al., 1977) identifies different "fragmentations," or sub-tasks, across all learned tasks. Once learned, the fragmentations are used to augment the state of the agent. Each sub-problem can be learned independently; when encountering a new task, much of the learning is already complete because the majority of sub-problems are unchanged. The fragmentations work with both a flat learner (i.e., TD) and an explicitly hierarchical learner to improve the jumpstart and total reward.

Probabilistic policy reuse (Fernandez and Veloso, 2006) also considers a distribution of tasks in which only the goal state differs, but is one of the most robust MTL methods in terms of appropriate source task selection. Although the method allows a single goal state to differ between the tasks, it requires that S, A, and T remain constant. If a newly learned policy is significantly different from existing policies, it is added to a policy library. When the agent is placed in a novel task, on every timestep, it can choose to: exploit a learned source task policy, exploit the current best policy for the target task, or randomly explore. If the agent has multiple learned policies in its library, it probabilistically selects between policies so that over time more useful policies will be selected more often. While this method allows for probabilistic mixing of the policies, it may be possible to treat the past policies as options which can be executed until some termination condition is met, similar to a number of previously discussed methods. By comparing the relative benefits of mixing past policies and treating them as options, it may be possible to better understand when each of the two approaches is most useful.

The idea of constructing an explicit policy library is likely to be useful in future TL research, particularly for agents that train on a number of source tasks that have large qualitative differences (and thus very different learned behaviors). Although other methods also separately record information from multiple source tasks (cf., Mehta et al., 2008; Lazaric, 2008), Fernandez and Veloso explicitly reason about the library. In addition to reasoning over the amount of information stored, as a function of number and type of source tasks, it will be useful to understand how many target task samples are needed to select the most useful source task(s).

Unlike probabilistic policy reuse, which selectively transfers information from a single source task, Tanaka and Yamamura (2003) gather statistics about *all* previous tasks and use this amalgamated knowledge to learn novel tasks faster. Specifically, the learner keeps track of the average

and the deviation of the action value for each (s,a) pair observed in all tasks. When the agent encounters a new task, it initializes the action-value function so that every (s,a) pair is set to the current average for that pair, which provides a benefit relative to uninformed initialization. As the agent learns the target task with Q-learning and prioritized sweeping,[7] the agent uses the standard deviation of states' Q-values to set priorities on TD backups. If the current Q-value is far from the average for that (s,a) pair, its value should be adjusted more quickly, since it is likely incorrect (and thus should be corrected before affecting other Q-values). Additionally, another term accounting for the variance within individual trials is added to the priority; Q-values that fluctuate often within a particular trial are likely wrong. Experiments show that this method, when applied to sets of discrete tasks with different transition functions, can provide significant improvement to jumpstart and total reward.

The next two methods consider how priors can be effectively learned by a Bayesian MTL agent. First, Sunmola and Wyatt (2006) introduce two methods that use instances from source tasks to set priors in a Bayesian learner. Both methods constrain the probabilities of the target task's transition function by using previous instances as a type of prior. The first method uses the working prior to generate possible models which are then tested against data in the target task. The second method uses a probability perturbation method in conjunction with observed data to improve models generated by the prior. Initial experiments show that the jumpstart and total reward can be improved if the agent has an accurate estimation of the prior distributions of the class from which the target is drawn. Second, Wilson et al. (2007) consider learning in a hierarchical Bayesian RL setting. Setting the prior for Bayesian models is often difficult, but in this work the prior may be transferred from previously learned tasks, significantly increasing the learning rate. Additionally, the algorithm can handle "classes" of MDPs, which have similar model parameters, and then recognize when a novel class of MDP is introduced. The novel class may then be added to the hierarchy and a distinct prior may be learned, rather than forcing the MDP to fit into an existing class. The location of the goal state and the parameterized reward function may differ between the tasks. Learning on subsequent tasks shows a clear performance improvement in total reward, and some improvement in jumpstart.

While Bayesian methods have been shown to be successful when transferring between classification tasks (Roy and Kaelbling, 2007), and in non-transfer RL (Dearden et al., 1999), only the two methods above use it in RL transfer. The learner's bias is important in all machine learning settings. However, Bayesian learning makes such bias explicit. Being able to set the bias through transfer from similar tasks may prove to be a very useful heuristic—we hope that additional transfer methods will be developed to initialize Bayesian learners from past tasks.

Walsh et al. (2006) observe that "deciding what knowledge to transfer between environments can be construed as determining the correct state abstraction scheme for a set of source [tasks] and then applying this compaction to a target [task]." Their suggested framework solves a set of MDPs, builds abstractions from the solutions, extracts relevant features, and then applies the feature-based abstraction function to a novel target task. A simple experiment using tasks with different state spaces and reward functions shows that the time to learn a target task is decreased by using MTL. Building upon their five defined types of state abstractions (as defined in Li et al. 2006), they give theoretical results showing that when the number of source tasks is large (relative to the differences

7. Prioritized sweeping (Moore and Atkeson, 1993) is an RL method that orders adjustments to the value function based on their "urgency," which can lead to faster convergence than when updating the value function in the order of visited states.

Citation	Allowed Task Differences	Source Task Selection	Task Mappings	Transferred Knowledge	Allowed Learners	TL Metrics
Different state variables and actions – no explicit task mappings: Section 6						
Konidaris and Barto (2006)	p	h	N/A	R	TD	j, tr
Konidaris and Barto (2007)	p	h	N/A	π_p	TD	j, tr
Banerjee and Stone (2007)	a, v	h	N/A	fea	TD	ap, j, tr
Guestrin et al. (2003)	#	h	N/A	Q	LP	j
Croonenborghs et al. (2007)	#	h	N/A	π_p	RRL	ap, j, tr
Ramon et al. (2007)	#	h	N/A	Q	RRL	ap, j, $tt^{\dagger}$, tr
Sharma et al. (2007)	#	h	N/A	Q	TD, CBR	j, tr

Table 5: This table reproduces the third group of methods from Table 1.

between the different tasks), four of the five types of abstractions are guaranteed to produce the optimal policy in a target task using Q-learning.

Similar to Walsh et al. (2006), Lazaric (2008) also discovers features to transfer. Rather than learning tasks sequentially, as in all the papers above, one could consider learning different tasks in parallel and using the shared information to learn the tasks better than if each were learned in isolation. Specifically, Lazaric (2008) learns a set of tasks with different reward functions using the batch method *Fitted Q-iteration* (Ernst et al., 2005). By leveraging a multi-task feature learning algorithm (Argyrious et al., 2007), the problem can be formulated as a joint optimization problem to find the best features and learning parameters across observed data in all tasks. Experiments demonstrate that this method can improve the total reward and can help the agent to ignore irrelevant features (i.e., features which do not provide useful information). Furthermore, since it may be possible to learn a superior representation, asymptotic performance may be improved as well, relative to learning tasks in isolation.

The work in this section, as summarized in the second section of Table 1, explicitly assumes that all MDPs an agent experiences are drawn from the same distribution. Different tasks in a single distribution could, in principal, have different state variables and actions, and future work should investigate when allowing such flexibility would be beneficial.

6. Transferring Task-Invariant Knowledge Between Tasks with Differing State Variables and Actions

This section, unlike the previous two, discusses methods that allow the source task and target task to have different state variables and actions (see Figure 8 and the methods in Table 5). These methods formulate the problem so that no explicit mapping between the tasks is needed. Instead the agent reasons over abstractions of the MDP that are invariant when the actions or state variables change.

For example, Konidaris and Barto (2006) have separated the standard RL problem into *agent-space* and *problem-space* representations. The agent-space is determined by the agent's capabilities, which remain fixed (e.g., physical sensors and actuators), although such a space may be non-Markovian.[8] The problem-space, on the other hand, may change between source and target

8. A standard assumption is that a task is Markovian, meaning that the probability distribution over next states is independent of the agent's state and action history. Thus, saving a history would not assist the agent when selecting actions, and it can consider each state in isolation.

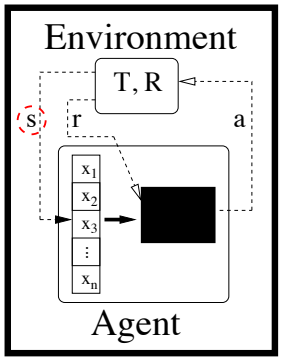


Figure 8: Methods in Section 6 are able to transfer between tasks with different state spaces. Although T, R, A, and the state variables may also technically change, the agent's internal representation is formulated so that they remain fixed between source and target tasks. MDP components with a dashed circle may change between the source task and target task.

problems and is assumed to be Markovian. The authors' method learns a shaping reward on-line in agent-space while learning a source task. If a later target task has a similar reward structure and action set, the learned shaping reward will help the agent achieve a jumpstart and higher total reward. For example, suppose that one of the agent's sensors measures the distance between it and a particular important state (such as a beacon located near the goal state). The agent may learn a shaping reward that assigns reward when the state variable describing its distance to the beacon is reduced, even in the absence of an environmental reward. The authors assume that there are no novel actions (i.e., actions which are not in the source task's problem-space) but any new state variables can be handled if they can be mapped from the novel problem-space into the familiar agent-space. Additionally, the authors acknowledge that the transfer must be between *reward-linked* tasks, where "the reward function in each environment consistently allocates rewards to the same types of interactions across environments." Determining whether or not a sequence of tasks meet this criterion is left for future work.

In later work (Konidaris and Barto, 2007), the authors assume knowledge of "pre-specified salient events," which make learning options tractable. While it may be possible to learn options without requiring such events to be specified, the paper focuses on how to use such options rather than option learning. Specifically, when the agent achieves one of these subgoals, such as unlocking a door or moving through a doorway, it may learn an option to achieve the event again in the future. As expected, problem-space options speed up learning a single task. More interesting, when the agent trains on a series of tasks, options in both agent-space and problem-space significantly increase the jumpstart and total reward in the target task (time spent learning the source task is discounted). The authors suggest that agent-space options will likely be more portable than problem-space options in cases where the source and target tasks are less similar—indeed, problem-space options will only be portable when source and target tasks are very similar.

In our opinion, agent- and problem-space are ideas that should be further explored as they will likely yield additional benefits. Particularly in the case of physical agents, it is intuitive that agent sensors and actuators will be static, allowing information to be easily reused. Task-specific items,

such as features and actions, may change, but should be faster to learn if the agent has already learned something about its unchanging agent-space.

If transfer is applied to game trees, changes in actions and state variables may be less problematic. Banerjee and Stone (2007) are able to transfer between games by focusing on this more abstract formulation. For instance, in experiments the learner identified the concept of a *fork*, a state where the player could win on the subsequent turn regardless of what move the opponent took next. After training in the source task, analyzing the source task data for such features, and then setting the value for a given feature based on the source task data, such features of the game tree were used in a variety of target tasks. This analysis focuses on the effects of actions on the game tree and thus the actions and state variables describing the source and target game can differ without requiring an inter-task mapping. Source task time is discounted, but jumpstart, total reward, and asymptotic performance are all improved via transfer. Although the experiments in the paper use only temporal difference learning, it is likely that this technique would work well with other types of learners.

Guestrin et al. (2003) examine a similar problem in the context of planning in what they term a *relational MDP*. Rather than learning a standard value function, an agent-centered value function for each *class* of agents is calculated in a source task, forcing all agents of a given class type to all have the same value function. However, these class value functions are defined so that they are independent of the number of agents in a task, allowing them to be directly used in a target task which has additional (or fewer) agents. No further learning is done in the target task, but the transferred value functions perform better than a handcoded strategy provided by the authors, despite having additional friendly and adversarial agents. However, the authors note that the technique will not perform well in heterogeneous environments or domains with "strong and constant interactions between many objects."

Relational Reinforcement Learning may also be used for effective transfer. Rather than reasoning about states as input from an agent's sensors, an RRL learner typically reasons about a state in propositional form by constructing first-order rules. The learner can easily abstract over specific object identities as well as the number of objects in the world; transfer between tasks with different number of objects is simplified. For instance, Croonenborghs et al. (2007) first learn a source task policy with RRL. The learned policy is used to create examples of state-action pairs, which are then used to build a relational decision tree. This tree predicts, for a given state, which action would be executed by the policy. Lastly, the trees are mined to produce *relational options*. These options are directly used in the target task with the assumption that the tasks are similar enough that no translation of the relational options is necessary. The authors consider three pairs of source/target tasks where relational options learned in the source directly apply to the target task (only the number of objects in the tasks may change), and learning is significantly improved in terms of jumpstart, total reward, and asymptotic performance.

Other work using RRL for transfer (Ramon et al., 2007) introduces the TGR algorithm, a relational decision tree algorithm. TGR incrementally builds a decision tree in which internal nodes use first-order logic to analyze the current state and where the tree's leaves contain action-values. The algorithm uses four tree-restructuring operators to effectively use available memory and increase sample efficacy. Both target task time and total time are reduced by first training on a simple source task and then on a related target task. Jumpstart, total reward, and asymptotic performance also appear to improve via transfer.

RRL is a particularly attractive formulation in the context of transfer learning. In RRL, agents can typically act in tasks with additional objects without reformulating their, although additional

training may be needed to achieve optimal (or even acceptable) performance levels. When it is possible to frame a domain of interest as an RRL task, transfer between tasks with different numbers of objects or agents will likely be relatively straightforward.

With motivation similar to that of RRL, some learning problems can be framed so that agents choose between high-level actions that function regardless of the number of objects being reasoned about. Sharma et al. (2007) combines case-based reasoning with RL in the *CAse-Based Reinforcement Learner* (CARL), a multi-level architecture includes three modules: a planner, a controller, and a learner. The tactical layer uses the learner to choose between high-level actions which are independent of the number of objects in the task. The cases are indexed by: high-level state variables (again independent of the number of objects in the task), the actions available, the Q-values of the actions, and the cumulative contribution of that case on previous timesteps. Similarity between the current situation and past cases is determined by Euclidean distance. Because the state variables and actions are defined so that the number of objects in the task can change, the source and target tasks can have different numbers of objects (in the example domain, the authors use different numbers of player and opponent troops in the source and target tasks). Time spent learning the source task is not counted, but the target task performance is measured in terms of jumpstart, *asymptotic gain* (a metric related to the improvement in average reward over learning), and *overall gain* (a metric based on the total reward accrued).

In summary, methods surveyed in this section all allow transfer between tasks with different state variables and actions, as well as transfer functions, state spaces, and reward functions. By framing the task in an agent-centric space, limiting the domain to game trees, or using a learning method that reasons about variable numbers of objects, knowledge can be transferred between tasks with relative ease because problem representations do not change from the learner's perspective. In general, not all tasks may be formulated so that they conform to the assumptions made by TL methods presented in this section.

7. Explicit Mappings to Transfer between Different Actions and State Representations

This section of the survey focuses on a set of methods which are more flexible than those previously discussed as they allow the state variables and available actions to differ between source and target tasks (see Table 6 and Figure 9). All methods in this section use inter-task mappings, enabling transfer between pairs of tasks that could not be addressed by methods in the previous section. Note that because of changes in state variables and actions, R, S, and T, all technically change as well (they are functions defined over actions and state variables). However, as we elaborate below, some of the methods allow for significant changes in reward functions between the tasks, while most do not.

In Taylor et al. (2007a), the authors assume that a mapping between the source and target tasks is provided to the learner. The learner first trains in a source task using a value-function-learning method. Before learning begins in the target task, every action-value for each state in the target task is initialized via learned source task values. This work experimentally demonstrates that value-function transfer can cause significant speedup by transferring between tasks that have different state variables and actions. Additionally, different methods for performing the value-function transfer are examined, different function approximators are successfully used, and multi-step transfer is demonstrated (i.e., transfer from task A to task B to task C). This TL method demonstrates that when

Citation	Allowed Task Differences	Source Task Selection	Task Mappings	Transferred Knowledge	Allowed Learners	TL Metrics
Different state variables and actions – inter-task mappings used: Section 7						
Taylor et al. (2007a)	a, v	h	sup	Q	TD	tt†
Taylor et al. (2007b)	a, v	h	sup	π	PS	tt†
Taylor et al. (2008b)	a, v	h	sup	I	MB	ap, tr
Torrey et al. (2005) Torrey et al. (2006)	a, r, v	h	sup	rule	TD	j, tr
Torrey et al. (2007)	a, r, v	h	sup	π_p	TD	j, tr
Taylor and Stone (2007b)	a, r, v	h	sup	rule	any/TD	j, tt†, tr

Table 6: This table reproduces the fourth group of methods from Table 1.

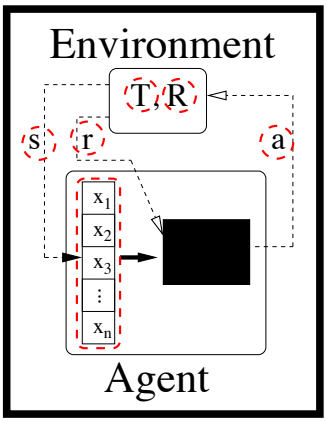


Figure 9: Methods in Section 7 focus on transferring between tasks with different state features, action sets, and possible reward functions (which, in turn, causes the state space and transition function to differ as well). As in previous figures, MDP components with a dashed circle may change between the source task and target task.

faced with a difficult task, it may be faster overall to first train on an artificial source task or tasks and then transfer the knowledge to the target task, rather than training on the target task directly. The authors provide no theoretical guarantees about their method's effectiveness, but hypothesize conditions under which their TL method will and will not perform well, and provide examples of when their method fails to reduce the training time via transfer.

In subsequent work, Taylor et al. (2007b) transfer entire policies between tasks with different state variables and actions, rather than action-value functions. A set of policies is first learned via a genetic algorithm in the source task and then transformed via inter-task mappings. Additionally, partial inter-task mappings are introduced, which may be easier for a human to intuit in many domains. Specifically, those actions and state variables in the target which have "very similar" actions and state variables in the source task are mapped, while novel state variables and actions in the target task are left unmapped. Policies are transformed using one of the inter-task mappings and then used to seed the learning algorithm in the target task. As in the previous work, this TL method can successfully reduce both the target task time and the total time.

Later, Taylor et al. (2008b) again consider pairs of tasks where the actions differ, the state variables differ, and inter-task mappings are available to the learner. In this work, the authors allow transfer between model-learning methods by transferring instances, which is similar in spirit to Lazaric (2008). Fitted R-MAX (Jong and Stone, 2007), an instance-based model-learning method capable of learning in continuous state spaces, is used as the base RL method, and source task instances are transferred into the target task to better approximate the target task's model. Experiments in a simple continuous domain show that transfer can improve the jumpstart, total reward, and asymptotic performance in the target task.

Another way to transfer is via learned *advice* or preferences. Torrey et al. (2005) automatically extract such advice from a source task by identifying actions which have higher Q-values than other available actions.[9] Such advice is mapped via human-provided inter-task mappings to the target task as preferences given to the target task learner. In this work, Q-values are learned via support vector regression, and then *Preference Knowledge Based Kernel Regression* (KBKR) (Maclin et al., 2005) adds the advice as soft constraints in the target, setting relative preferences for different actions in different states. The advice is successfully leveraged by the target task learner and decreases the target task learning time, even when the source task has different state variables and actions. Additionally, the reward structure of the tasks may differ substantially: their experiments use a source task whose reward is an unbounded score based on episode length, while the target task's reward is binary, depending on if the agents reached a goal state or not. Source task time is discounted and the target task learning is improved slightly in terms of total reward and asymptotic performance.

Later work (Torrey et al., 2006) improves upon this method by using *inductive logic programming* (ILP) to identify *skills* that are useful to the agent in a source task. A trace of the agent in the source task is examined and both positive and negative examples are extracted. Positive and negative examples are identified by observing which action was executed, the resulting outcome, the Q-value of the action, and the relative Q-value of other available actions. Skills are extracted using the ILP engine Aleph (Srinivasan, 2001) by using the F_1 score (the harmonic mean of precision and recall). These skills are then mapped by a human into the target task, where they improve learning via KBKR. Source task time is not counted towards the target task time, jumpstart may be improved, and the total reward is improved. The source and target tasks again differ in terms of state variables, actions, and reward structure. The authors also show how human-provided advice may be easily incorporated in addition to advice generated in the source task. Finally, the authors experimentally demonstrate that giving bad advice to the learner is only temporarily harmful and that the learner can "unlearn" bad advice over time, which may be important for minimizing the impact of negative transfer.

Torrey et al. (2007) further generalize their technique to transfer *strategies*, which may require composing several skills together, and are defined as a finite-state machine (FSM). The *structure learning* phase of their algorithm analyzes source task data to find sequences of actions that distinguish between successful and unsuccessful games (e.g., whether or not a goal was reached), and composes the actions into a FSM. The second phase, *ruleset learning*, learns when each action in the strategy should be taken based on state features, and when the FSM should transition to the next state. Experience in the source task is again divided into positive and negative sequences for Aleph. Once the strategies are re-mapped to the target task via a human-provided mapping, they are used to *demonstrate* a strategy to the target task learner. Rather than explore randomly, the target

9. While this survey focuses on automatically learned knowledge in a source task, rather than human-provided knowledge, Torrey et al. (2005) show that both kinds of knowledge can be effectively leveraged.

task learner always executes the transferred strategies for the first 100 episodes and thus learns to estimate the Q-values of the actions selected by the transferred strategies. After this demonstration phase, the learner chooses from the MDP's actions, not the high-level strategies, and can learn to improve on the transferred strategies. Experiments demonstrate that strategy transfer significantly improves the jumpstart and total reward in the target task when the source and target tasks have different state variables and actions (source task time is again discounted).

Similar to strategy transfer, Taylor and Stone (2007b) learn *rules* with RIPPER (Cohen, 1995) that summarize a learned source task policy. The rules are then transformed via handcoded inter-task mappings so that they could apply to a target task with different state variables and actions. The target task learner may then bootstrap learning by incorporating the rules as an extra action, essentially adding an ever-present option "take the action suggested by the source task policy," resulting in an improved jumpstart and total reward. By using rules as an intermediary between the two tasks, the authors argue that the source and target tasks can use different learning methods, effectively de-coupling the two learners. Similarities with Torrey et al. (2007) include a significant improvement in initial performance and no provision to automatically handle scale differences.[10] The methods differ primarily in how advice is incorporated into the target learner and the choice of rule learner.

Additionally, Taylor and Stone (2007b) demonstrated that *inter-domain* transfer is possible. The two source tasks in this paper were discrete, fully observable, and one was deterministic. The target task, however, had a continuous state space, was partially observable, and had stochastic actions. Because the source tasks required orders of magnitude less time, the total time was roughly equal to the target task time. Our past work has used the term "inter-domain transfer" for transfer between qualitatively different domains, such as between a board game and a soccer simulation. However, this term is not well defined, or even agreed upon in the community. For instance, Swarup and Ray (2006) use the term "cross-domain transfer" to describe the reuse of a neural network structure between classification tasks with different numbers of boolean inputs and a single output. However, our hope is that researchers will continue improve transfer methods so that they may usefully transfer from very dissimilar tasks, similar to the way that humans may transfer high level ideas between very different domains.

This survey has discussed examples of of low- and high-level knowledge transfer. For instance, learning general rules or advice may be seen as relatively high level, whereas transferring specific Q-values or observed instances is quite task-specific. Our intuition is that higher-level knowledge may be more useful when transferring between very dissimilar tasks. For instance, it is unlikely that Q-values learned for a checkers game will transfer to chess, but the concept of a fork may transfer well. This has not been definitely shown, however, nor is there a quantitative way to classify knowledge in terms of low- or high-level. We hope that future work will confirm or disconfirm this hypothesis, as well as generate guidelines as to when different types of transferred knowledge is most appropriate.

All methods in this section use some type of inter-task mapping to allow transfer between MDPs with very different specifications. While these results show that transfer can provide a significant benefit, they presuppose that the mappings are provided to the learner. The following section considers methods that work to autonomously learn such inter-task mappings.

10. To our knowledge, there is currently no published method to automatically scale rule constants. Such scaling would be necessary if, for instance, source task distances were measured in feet, but target task distances were measured in meters.

Citation	Allowed Task Differences	Source Task Selection	Task Mappings	Transferred Knowledge	Allowed Learners	TL Metrics
Learning inter-task mappings: Section 8						
Kuhlmann and Stone (2007)	a, v	h	T	Q	TD	j, tr
Liu and Stone (2006)	a, v	h	T	N/A	all	N/A
Soni and Singh (2006)	a, v	h	M_a, sv_g, exp	N/A	all	ap, j, tr
Talvitie and Singh (2007)	a, v	h	M_a, sv_g, exp	N/A	all	j
Taylor et al. (2007b)*	a, v	h	sv_g, exp	N/A	all	tt†
Taylor et al. (2008c)	a, v	h	exp	N/A	all	j, tr

Table 7: This table reproduces the group of inter-task learning methods from Table 1.

8. Learning Task Mappings

The transfer algorithms considered thus far have assumed that a hand-coded mapping between tasks was provided, or that no mapping was needed. In this section we consider the less-well explored question of how a mapping between tasks can be learned, such that source task knowledge may be exploited in a novel target task with different state variables and actions (see Figure 10 and the final group in Table 1). Note that in this section, all but one of the methods have N/A for transfer method—with the exception of Kuhlmann and Stone (2007), the papers covered in this section introduce mapping-learning methods and then use existing methods to validate the mapping efficacy.

One current challenge of TL research is to reduce the amount of information provided to the learner about the relationship between the source and target tasks. If a human is directing the learner through a series of tasks, the similarities (or analogies) between the tasks will likely be provided by the human's intuition. If transfer is to succeed in an autonomous setting, however, the learner must first determine how (and whether) two tasks are related, and only then may the agent leverage its past knowledge to learn in a target task. Learning task relationships is critical if agents are to transfer without human input, either because the human is outside the loop, or because the human is *unable* to provide similarities between tasks. Methods in this section differ primarily in what information must be provided. At one end of the spectrum, Kuhlmann and Stone (2007) assume that a complete description of R, S, and T are given, while at the other, Taylor et al. (2008c) learn the mapping exclusively from experience gathered via environmental interactions.

Given a complete description of a game (i.e., the full model of the MDP), Kuhlmann and Stone (2007) analyze the game to produce a *rule graph*, an abstract representation of a deterministic, full information game. A learner first trains on a series of source task games, storing the rule graphs and learned value functions. When a novel target task is presented to the learner, it first constructs the target task's rule graph and then attempts to find a source task that has an isomorphic rule graph. The learner assumes that a transition function is provided and uses value-function-based learning to estimate values for *afterstates* of games. Only state variables need to be mapped between source and target tasks, and this is exactly the mapping found by graph matching. For each state in the target task, initial Q-values are set by finding the value of the corresponding state in the source task. Three types of transfer are considered: direct, which copies afterstate values over without modification; inverse, which accounts for a reversed goal or switched roles; and average, with copies the average

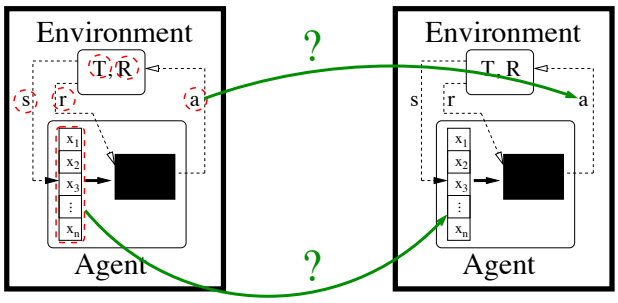


Figure 10: Section 8 presents methods to learn the relationship between tasks with different state variables and actions. As in previous figures, MDP components with a dashed circle may change between the source task and target task.

of a set of Q-values and can be used for boards with different sizes. Source task time is ignored but jumpstart and total reward can both be improved in the target task.

The previous work assumes full knowledge of a transition function. A more general approach could assume that the agent has only a qualitative understanding of the transition function. For instance, *qualitative dynamic Bayes networks* (QDBNs) (Liu and Stone, 2006), summarize the effects of actions on state variables but are not precise (for instance, they could not be used as a generative model for planning). If QDBNs are provided to an agent, a graph mapping technique can automatically find a mapping between actions and state variables in two tasks with relatively little computational cost. The authors show that mappings can be learned autonomously, effectively enabling value function transfer between tasks with different state variables and actions. However, it remains an open question as to whether or not QDBNs are learnable from experience, rather than being hand-coded.

The next three methods assume knowledge about how state variables are used to describe objects in a multi-player task. For instance, an agent may know that a pair of state variables describe "distance to teammate" and "distance from teammate to marker," but the agent is not told *which* teammate the state variables describe. First, Soni and Singh (2006) supply an agent with a series of possible state transformations and an inter-task action mapping. There is one such transformation, X, for every possible mapping of target task variables to source task variables. After learning the source task, the agent's goal is to learn the correct transformation: in each target task state s, the agent can randomly explore the target task actions, or it may choose to take the action $\pi_{source}(X(s))$. This method has a similar motivation to that of Fernandez and Veloso (2006), but here the authors are learning to select between possible mappings rather than possible previous policies. Over time the agent uses Q-learning to select the best state variable mapping as well as learn the action-values for the target task. The jumpstart, total reward, and asymptotic performance are all slightly improved when using this method, but its efficacy will be heavily dependent on the number of possible mappings between any source and target task.

Second, *AtEase* (Talvitie and Singh, 2007) also generates a number of possible state variable mappings. The action mapping is again assumed and the target task learner treats each of the possible mappings as an arm on a multi-armed bandit (Bellman, 1956). The authors prove their algorithm learns in time proportional to the number of possible mappings rather than the size of the problem: "in time polynomial in T, [the algorithm] accomplishes an actual return close to the

asymptotic return of the best expert that has mixing time at most T." This approach focuses efficient selection of a proposed state variable mappings and does not allow target task learning.

Third, these assumptions are relaxed slightly by Taylor et al. (2007b), who show that it is possible to learn both the action and state variable mapping simultaneously by leveraging a classification technique, although it again relies on the pre-specified state variable groupings (i.e., knowing that "distance to teammate" refers to a teammate, but not which teammate). Action and state variable classifiers are trained using recorded source task data. For instance, the source task agent records s_{source}, a_{source}, s'_{source} tuples as it interacts with the environment. An action classifier is trained so that $C(s_{source,object}, s'_{source,object}) = a_{source}$ for each object present in the source task. Later, the target task agent again records s_{target}, a_{target}, s'_{target} tuples. Then the action classifier can again be used for to classify tuples for every target task object: $C(s_{target,object}, s'_{target,object}) = a_{source}$, where such a classification would indicate a mapping between a_{target} and a_{source}. Relatively little data is needed for accurate classification; the number of samples needed to learn in the target task far outweighs the number of samples used by the mapping-leaning step. While the resulting mappings are not always optimal for transfer, they do serve to effectively reduce target task training time as well as the total training time.

The MASTER algorithm (Taylor et al., 2008c) was designed to further relax the knowledge requirements of Taylor et al. (2007b): no state variable groupings are required. The key idea of MASTER is to save experienced source task instances, build an approximate transition model from a small set of experienced target task instances, and then test possible mappings offline by measuring the prediction error of the target-task models on source task data. This approach is sample efficient at the expense of high computational complexity, particularly as the number of state variables and actions increase. The method uses an exhaustive search to find the inter-task mappings that minimize the prediction error, but more sophisticated (e.g., heuristic) search methods could be incorporated. Experiments show that the learned inter-task mappings can successfully improve jumpstart and total reward. A set of experiments also shows how the algorithm can assist with source task selection by selecting the source task which is best able to minimize the offline prediction error. The primary contribution of MASTER is to demonstrate that autonomous transfer is possible, as the algorithm can learn inter-task mappings autonomously, which may then be used by any of the TL methods discussed in the previous section of this survey (Section 7).

In summary, this last section of the survey has discussed several methods able to learn inter-task mappings with different amounts of data. Although all make some assumptions about the amount of knowledge provided to the learner or the similarity between source and target tasks, these approaches represent an important step towards achieving fully autonomous transfer.

The methods in the section have been loosely ordered in terms of increasing autonomy. By learning inter-task mappings, these algorithms try to enable a TL agent to use past knowledge on a novel task without human intervention, even if the state variables or actions change. However, the question remains whether fully autonomous transfer would ever be useful in practice. Specifically, if there are no restrictions on the type of target task that could be encountered, why would one expect that past knowledge (a type of bias) would be useful when learning an encountered task, or even on the majority of tasks that could be encountered? This question is directly tied to the ability of TL algorithms to recognize when tasks are similar and when negative transfer may occur, both of which are discussed in more detail in the following section.

9. Open Questions

Although transfer learning in RL has made significant progress in recent years, there are still a number of open questions to be addressed. This section presents a selection of questions that we find particularly important. Section 9.1 discusses ways in which methods in the survey could potentially be extended and serves to highlight some of the methods most promising for future work. Section 9.2 then discusses the problem of negative transfer, currently one of the most troubling open questions. Lastly, Section 9.3 presents a set of possible research directions that the authors' believe will be most beneficial to the field of TL.

9.1 Potential Enhancements

One apparent gap in our taxonomy is a dearth of model-learning methods. Because model-learning algorithms are often more sample efficient than model-free algorithms, it is likely that TL will have a large impact on sample complexity when coupled with such efficient RL methods. Moreover, when a full model of the environment is learned in a source task, it may be possible for the target task learner to explicitly reason about how to refine or extend the model as it encounters disparities between it and the target task.

As mentioned in Section 5, transfer is an appealing way to set priors in a Bayesian setting. When in a MTL setting, it may be possible to accurately learn priors over a distribution of tasks, enabling a learner to better avoid negative transfer. One of the main benefits of transfer learning is the ability to bias learners so that they may find better solutions with less data; making these biases explicit through Bayesian priors may allow more efficient (and human-understandable) transfer methods. While there will likely be difficulties associated with scaling up current methods to handle complex tasks, possibly with a complex distribution hierarchy, it seems like Bayesian methods are particularly appropriate for transfer.

The idea of automatically modifying source tasks (cf., RTP Sherstov and Stone 2005, and suggested by Kuhlmann and Stone 2007) has not yet been widely adopted. However, such methods have the potential to improving transfer efficacy in settings where the target task learning performance is paramount. By developing methods that allow training on a sequence of automatically generated variations, TL agents may be able to train autonomously and gain experience that is exploitable in a novel task. Such an approach would be particularly relevant in the multi-task learning setting where the agent could leverage some assumptions about the distribution of the target task(s) it will see in the future.

None of the transfer methods in this survey are able to explicitly take advantage of any knowledge about changes in the reward function between tasks, and it may be particularly easy for humans to identify qualitative changes in reward functions. For example, if it was known that the target task rewards were twice that of the source task, it is possible that value-function methods may be able to automatically modify the source task value function with this background knowledge to enhance learning. As a second example, consider a pair of tasks where the goal state were moved from one edge of the state space to the opposite edge. While the learned transition information could be reused, the policy or value-function would need to be significantly altered to account for the new reward function. It is possible that inter-task mappings could be extended to account for changes in R between tasks, in addition to changes in A and in state variables.

Ideas from *theory revision* (Ginsberg, 1988) (also *theory refinement*) may help inform the automatic construction of inter-task mappings. For example, many methods initialize a target task

agent to have Q-values similar to those in the source task agent. Transfer is likely to be successful (Taylor et al., 2007a) if the target task Q-values are close enough to the optimal Q-values that learning is improved, relative to not using transfer. There are also situations where a *syntactic* change to the knowledge would produce better transfer. For instance, if the target task's reward function were the inverse of the source task function, direct transfer of Q-values would be far from optimal. However, a TL algorithm that could recognize the inverse relationship may be able to use the source task knowledge more appropriately (such as initializing its behavior so that $\pi_{target}(s_{target}) \neq \pi_{source}(\chi_X(s_{target}))$.

Given a successful application of transfer, there are potentially two distinct benefits for the agent. First, transfer may help improve the agent's exploration so that it discovers higher-valued states more quickly. Secondly, transfer can help bias the agent's internal representation (e.g., its function approximator) so that it may learn faster. It will be important for future work to better distinguish between these two effects; decoupling the two contributions should allow for a better understanding of TL's benefits, as well as provide avenues for future improvements.

Of the thirty-four transfer methods discussed, only five (Tanaka and Yamamura, 2003; Sunmola and Wyatt, 2006; Ferguson and Mahadevan, 2006; Lazaric, 2008; Wilson et al., 2007) attempt to discover internal learning parameters (e.g., appropriate features or learning rate) so that future tasks in the same domain may be learned more efficiently. It is likely that other "meta-learning" methods could be useful. For instance, it may be possible to learn to use an appropriate function approximator, an advantageous learning rate, or even the most appropriate RL method. Although likely easier to accomplish in a MTL setting, such meta-learning may also be possible in transfer, given sufficiently strong assumptions about task similarity. Multiple heuristics regarding the best way to select RL methods and learning parameter settings for a particular domain exist, but typically such settings are chosen in an ad hoc manner. Transfer may be able to assist when setting such parameters, rather than relying on human intuition.

Section 8 discussed methods that learned an inter-task mapping, with the motivation that such a mapping could enable autonomous transfer. However, it is unclear if fully autonomous TL is realistic in an RL setting, or indeed is useful. In the majority of situations, a human will be somewhere in the loop and full autonomy is not necessary. Instead, it could be that mappings may be learned to *supplement* a human's intuition regarding appropriate mappings, or that a set of learned mappings could be proposed and then one selected by a human. It would be worthwhile to define realistic scenarios when fully autonomous transfer will be necessary, or to instead specify how (limited) human interaction will be coupled with mapping-learning methods.

Lastly, we hope that the idea of task-invariant knowledge will be extended. Rather than learning an appropriate representation across tasks, agent-space (Konidaris and Barto, 2007) and RRL techniques attempt to discover knowledge about the agent or the agent's actions which can be directly reused in novel tasks. The better techniques can successfully compartmentalize knowledge, separating what will usefully transfer and what will not will not, the easier it will be to achieve successful transfer without having to un-learn irrelevant biases.

9.2 Negative Transfer

The majority of TL work in the literature has concentrated on showing that a particular transfer approach is plausible. None, to our knowledge, has a well-defined method for determining *when* an approach will fail according to one or more metrics. While we can say that it is possible to improve

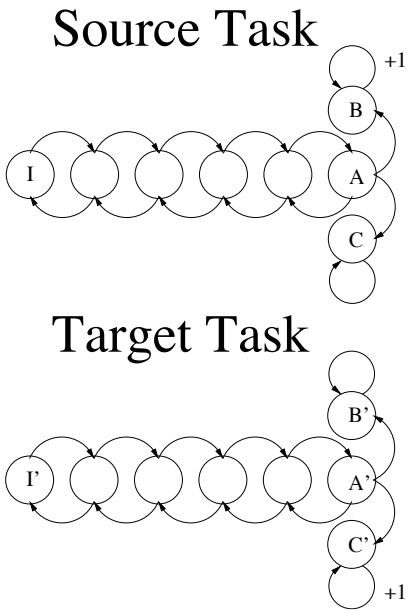


Figure 11: This figure depicts a pair of tasks that are likely to result in negative transfer for TL methods.

learning in a target task faster via transfer, we cannot currently decide if an arbitrary pair of tasks are appropriate for a given transfer method. Therefore, transfer may produce incorrect learning biases and result in negative transfer.

Methods such as MASTER (Taylor et al., 2008c), which can measure task similarity via model prediction error, or region transfer (Lazaric, 2008), which examines the similarity of tasks at a local level rather than at a per-task level, can help assist when deciding if the agent should transfer or what the agent should transfer. However, neither method provides any theoretical guarantees about its effectiveness.

As an example of why it is difficult to define a metric for task similarity, consider the pair of tasks shown in Figure 11, which are extremely similar, but where direct transfer of a policy or action-value function will be detrimental. The source task in Figure 11 (top) is deterministic and discrete. The agent begins in state I and has one action available: `East`. Other states in the "hallway" have two applicable actions: `East` and `West`, except for state A, which also has the actions `North` and `South`. Once the agent executes `North` or `South` in state A, it will remain in state B or C (respectively) and continue self-transitioning. No transition has a reward, except for the self-transition in state B.

Now consider the target task in Figure 11 (bottom), which is the same as the source task, except that the self-transition from C′ is the only rewarded transition in the MDP. $Q^\star$(I′,`East`) in the target task (the optimal action-value function, evaluated at the state I′) is the same as $Q^\star$(I, `East`) in the source task. Indeed, the optimal policy in the target task differs at only a single state, A′, and the optimal action-value functions differ only at states A′, B′, and C′.

One potential method for avoiding negative transfer is to leverage the ideas of *bisimulation* (Milner, 1982). Ferns et al. (2006) point out that:

> In the context of MDPs, bisimulation can roughly be described as the largest equivalence relation on the state space of an MDP that relates two states precisely when for every action, they achieve the same immediate reward and have the same probability of transitioning to classes of equivalent states. This means that bisimilar states lead to essentially the same long-term behavior.

However, bisimulation may be too strict because states are either equivalent or not, and may be slow to compute in practice. The work of Ferns et al. (2005, 2006) relaxes the idea of bisimulation to that of a (pseudo)metric that can be computed much faster, and gives a similarity measure, rather than a boolean. It is possible, although not yet shown, that bisimulation approximations can be used to discover regions of state space that can be transferred from one task to another, or to determine how similar two tasks are *in toto*. In addition to this, or perhaps because of it, there are currently no methods for automatically *constructing* a source task given a target task.[11]

Homomorphisms (Ravindran and Barto, 2002) are a different abstraction that can define transformations between MDPs based on transition and reward dynamics, similar in spirit to inter-task mappings, and have been used successfully for transfer (Soni and Singh, 2006). However, discovering homomorphisms is NP-hard (Ravindran and Barto, 2003a) and homomorphisms are generally supplied to a learner by an oracle. While these two theoretical frameworks may be able to help avoid negative transfer, or determine when two tasks are "transfer compatible," significant work needs to be done to determine if such approaches are feasible in practice, particularly if the agent is fully autonomous (i.e., is not provided domain knowledge by a human) and is not provided a full model of the MDP.

9.3 New Directions

As suggested above, TL in RL domains is one area of machine learning where the empirical work has outpaced the theoretical. While there has been some work on the theory of transfer between classification tasks (cf., Baxter, 2000; Ben-David and Borbely, 2008), such analyses do not directly apply to RL settings. To our knowledge, there is only a single work analyzing the theoretical properties of transfer in RL (Phillips, 2006), where the authors use the Kantorovich and full models of two MDPs to calculate how well an optimal policy in one task will perform in a second task. Unfortunately, this calculation of policy performance may require more computation than directly learning in the target task. There is considerable room, and need for, more theoretical work in RL (cf., Bowling and Veloso, 1999). For example:

1. Provides guarantees about whether a particular source task can improve learning in a target task (given a particular type of knowledge transfer).

2. Correlates the amount of knowledge transferred (e.g., the number of samples) with the improvement in the source task.

3. Defines what an optimal inter-task mapping is, and demonstrates how transfer efficacy is impacted by the inter-task mapping used.

11. We distinguish this idea from Sherstov and Stone's 2005 approach. Their paper shows it is possible to construct source task perturbations and then allow an agent to spend time learning the set of tasks to attempt to improve learning on an (unknown) source task. Instead, it may be more effective to tailor a source task to a specific target task, effectively enabling an agent to reduce the total number of environmental interactions needed to learn.

The remainder of this section suggests other open areas.

Concept drift (Widmer and Kubat, 1996) in RL has not been directly addressed by any work in this survey. The idea of concept drift is related to a non-stationary environment: at certain points in time, the environment may change arbitrarily. As Ramon et al. (2007) note, "for transfer learning, it is usually known when the context change takes place. For concept drift, this change is usually unannounced." Current on-line learning methods may be capable of handling such changes by continually learning. However, it is likely that RL methods developed specifically to converge to a policy and then re-start learning when the concept changes will achieve higher performance, whether such drift is announced or unannounced.

Another question no work in this survey directly addresses is how to determine the optimal amount of source task training to minimize the target task training time or total training time. If the source task and target task were identical, the goal of reducing the target task training time would be trivial (by maximizing the source task training time) and the goal of minimizing total training time would be impossible. On the other hand, if the source task and target task were unrelated, it would be impossible to reduce the target task training time through transfer and the total training time would be minimized by not training in the source task at all. It is likely that a calculation or heuristic for determining the optimal amount of source task training time will have to consider the structure of the two tasks, their relationship, and what transfer method is used. This optimization becomes even more difficult in the case of multi-step transfer, as there are two or more tasks that can be trained for different amounts of time.

Transfer methods in this survey have used source task knowledge in many forms to better learn in a target task. However, none explicitly account for scaling differences between the two tasks. For instance, if a source task measured distance in meters and the target task measured distance in inches, constants would have to be updated manually rather than learned.

Another question not addressed is how to best explore in a source task if the explicit purpose of the agent is to speed up learning in a target task. One could imagine that a non-standard learning or exploration strategy may produce better transfer results, relative to standard strategies. For instance, it may be better to explore more of the source task's state space than to learn an accurate action-value function for only part of the state space. While no current TL algorithms take such an approach, there has been some work on the question of learning a policy that is exploitable (without attempt to maximize the on-line reward accrued while learning) in non-transfer contexts (Şimşek and Barto, 2006).

Similarly, instead of always transferring information from the end of learning in the source task, an agent that knows its information will be used in a target task may decide to record information to transfer partway through training in the source task. For instance Taylor et al. (2007b) showed that transfer may be more effective when using policies trained for less time in the source task than when using those trained for more time. Although others have also observed similar behavior Mihalkova and Mooney (2008), the majority of work shows that increased performance in the source task is correlated with increased target task performance. Understanding how and why this effect occurs will help determine the most appropriate time to transfer information from one task to another.

We now present four possibilities for extending the current RL transfer work to different learning settings in which transfer has not been successfully applied.

- First, although two of the papers (Banerjee and Stone, 2007; Kuhlmann and Stone, 2007) in this survey have examined extensive games, none consider repeated normal form games or stochastic games (Shapley, 1953). For instance, one could consider learning how to play

against a set of opponents so that when a new opponent is introduced, the learner may quickly adapt one of its previous strategies rather than completely re-learning a strategy. Another option would be for an agent to learn how to play one game and then transfer the knowledge to a different stochastic game. Due to similarities between RL and these two game playing settings, transfer methods described in this survey may be applied with relatively little modification.

- A second possibility for extending transfer is into the realm of partially observable MDPs (POMDPs). It may possible to learn a source POMDP and then use knowledge gained to heuristically speed up planning in a target POMDP. Additionally, because it is typically assumed that POMDP planners are given a complete and accurate model of a task, it may be possible to analytically compare source and target tasks before learning in order to determine if transfer would be beneficial, and if so, how best to use the past knowledge.

- Third, multi-agent MDP and POMDP learners may also be able to successfully exploit transfer. None of the work surveyed in this article focuses on explicit multi-agent learning (i.e., learning over the joint action space, or in an (adaptive) adversarial setting, as in Stone and Veloso 2000), but it is likely existing methods may be extended to the cooperative multi-agent setting. For instance, when formulating a problem as an MMDP or DEC-MDP, the agents must either reason over a joint action space or explicitly reason about how their actions affect others. It may be possible for agents to learn over a subset of actions first, and then gradually add actions (or joint actions) over time, similar to transferring between tasks with different action sets. The need for such speedups is particularly critical in distributed POMDPs, as solving them optimally as been shown to be NEXP-Complete (Bernstein et al., 2002). Transfer is one possible approach to making such problems more tractable, but to our knowledge, no such methods have yet been proposed.

- Fourth, as mentioned in Section 3.3, MTL methods in RL consider a sequence of tasks that are drawn sequentially from the same distribution. However, in supervised learning, multi-task learning typically involves learning multiple tasks *simultaneously*. There may be contexts in which an agent must learn multiple tasks concurrently, such as in hierarchical RL or when the agent has multiple reward functions or goals. Fully specifying such a scenario, and extending MTL methods to encompass this setting, could bring additional tools to RL researchers and help move TL in RL closer to TL in classification.

Lastly, in order to better evaluate TL methods, it would be helpful to have a standard set of domains and metrics. Ideally there would be a domain-independent metric for transfer learning, but it is unclear that such a metric can exist (see Section 2). Furthermore, it is unclear what *optimal transfer* would mean, but would likely depend on the scenario considered. Classification and regression have long benefited from standard metrics, such as precision and recall, and it is likely that progress in transfer will be likewise enhanced once standard metrics are agreed upon.

Standard test sets, such as the Machine Learning Repository at the University of California, Irvine (Asuncion and Newman, 2007), have also assisted the growth and progress of supervised learning, but there are currently no equivalents for RL. Furthermore, while there are some standard data sets for for transfer learning in classification,[12] none exist for transfer in RL. While there is

12. Found at `http://multitask.cs.berkeley.edu`.

some work in the RL community to standardize on a common interface and set of benchmark tasks (Tanner et al., 2008; Whiteson et al., 2008), no such standardization has been proposed for the transfer learning in RL community. Even in the absence of such a framework, we suggest that it is important for authors working in this area to:

- Clearly specify the setting: Is the source task learning time discounted? What assumptions are made about the relationship between the source target and target task?
- Evaluate the algorithm with a number of metrics: No one metric captures all possible benefits from transfer.
- Empirically or theoretically compare the performance of novel algorithms: To better evaluate novel algorithms, existing algorithms should be compared using standard metrics on a single task task.[13]

As discussed in Section 2.1, we do not think that TL for RL methods can be strictly ordered in terms of efficacy, due to the many possible goals of transfer. However, by standardizing on reporting methodology, TL algorithms can be more easily compared, making it easier to select an appropriate method in a given experimental setting.

Our hope is that TL questions, such as those presented in this section, will be addressed in the near future; our expectation is that transfer learning will become an increasingly powerful tool for the machine learning community.

Acknowledgments

We would like to thank Cynthia Matuszek and the anonymous reviewers for helpful comments and suggestions over multiple revisions. This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-0615104), DARPA (FA8750-05-2-0283 and FA8650-08-C-7812), the Federal Highway Administration (DTFH61-07-H-00030), and General Motors.

References

Agnar Aamodt and Enric Plaza. Case-based reasoning: foundational issues, methodological variations, and system approaches, 1994.

Pieter Abbeel and Andrew Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*, pages 1–8, 2005.

13. One of the difficulties inherent in this proposal is that small variations in domain implementation may result in very different learning performances. While machine learning practitioners are able to report past results verbatim when using the same data set, many RL domains used in papers are not released. In order to compare with past work, RL researchers must reimplement, tune, and test past algorithms to compare with their algorithm on their domain implementation.

David Andre and Stuart J. Russell. State abstraction for programmable reinforcement learning agents. In *Proc. of the Eighteenth National Conference on Artificial Intelligence*, pages 119–125, 2002.

Andreas Argyrious, Theodoros Evgenion, and Massimiliano Pontil. Multitask reinforcement learning on the distribution of MDPs. *Machine Learning*, 2007.

Minoru Asada, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda. Vision-based behavior acquisition for a shooting robot by using a reinforcement learning. In *Proceedings of IAPR/IEEE Workshop on Visual Behaviors-1994*, pages 112–118, 1994.

Mehran Asadi and Manfred Huber. Effective control knowledge transfer through learning skill and representation hierarchies. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2054–2059, 2007.

Authur Asuncion and David J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Christopher G. Atkeson and Juan C. Santamaria. A comparison of direct and model-based reinforcement learning. In *Proceedings of the 1997 International Conference on Robotics and Automation*, 1997.

Bikramjit Banerjee and Peter Stone. General game learning using knowledge transfer. In *The 20th International Joint Conference on Artificial Intelligence*, pages 672–677, January 2007.

Bikramjit Banerjee, Yaxin Liu, and G. Michael Youngblood. ICML workshop on "Structural knowledge transfer for machine learning", June 2006.

Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

Jonathan Baxter and Peter L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

Richard E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Richard E. Bellman. A problem in the sequential design of experiments. *Sankhya*, 16:221–229, 1956.

Shai Ben-David and Reba Schuller Borbely. A notion of task relatedness yielding provable multiple-task learning guarantees. *Machine Learning*, 73:273–287, 2008.

Darrin C. Bentivegna, Christopher G. Atkeson, and Gordon Cheng. Learning from observation and practice using primitives. In *AAAI 2004 Fall Symposium on Real-life Reinforcement Learning*, October 2004.

Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27 (4):819–840, November 2002.

Michael H. Bowling and Manuela M. Veloso. Bounding the suboptimality of reusing subproblem. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1340–1347, San Francisco, CA, USA, 1999.

James L. Carroll and Kevin Seppi. Task similarity measures for transfer in reinforcement learning task libraries. *Proceedings of 2005 IEEE International Joint Conference on Neural Networks*, 2: 803–808, 2005.

Rich Caruana. Learning many related tasks at the same time with backpropagation. In *Advances in Neural Information Processing Systems 7*, pages 657–664, 1995.

Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

Dongkyu Choi, Tolgo Konik, Negin Nejati, Chunki Park, and Pat Langley. Structural transfer of cognitive skills. In *Proceedings of the Eighth International Conference on Cognitive Modeling*, 2007.

William W. Cohen. Fast effective rule induction. In *International Conference on Machine Learning*, pages 115–123, 1995.

Marco Colombetti and Marco Dorigo. Robot shaping: developing situated agents through learning. Technical Report TR-92-040, International Computer Science Institute, Berkeley, CA, 1993.

Robert H. Crites and Andrew G. Barto. Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 1017–1023, Cambridge, MA, 1996. MIT Press.

Tom Croonenborghs, Kurt Driessens, and Maurice Bruynooghe. Learning relational options for inductive transfer in relational reinforcement learning. In *Proceedings of the Seventeenth Conference on Inductive Logic Programming*, 2007.

DARPA. Transfer learning proposer information pamphlet, BAA #05-29, 2005.

Thomas Dean and Robert Givan. Model minimization in Markov decision processes. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 106–111, 1997.

Richard Dearden, Nir Friedman, and David Andre. Model based Bayesian exploration. In *Proceedings of the 1999 Conference on Uncertainty in Artificial Intelligence*, pages 150–159, 1999.

AArthur Dempster, Nan Laird, and Donald Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Soc. Set. B (methodological)*, 39:1–38, 1977.

Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

Chris Drummond. Accelerating reinforcement learning by composing solutions of automatically identified subtasks. *Journal of Artificial Intelligence Research*, 16:59–104, 2002.

Saso Dzeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. *Machine Learning*, 43(1/2):5–52, April 2001.

Tom Erez and William D. Smart. What does shaping mean for computational reinforcement learning? In *Proceedings of the Seventh IEEE International Conference on Development and Learning*, pages 215–219, 2008.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

Kimberly Ferguson and Sridhar Mahadevan. Proto-transfer learning in Markov decision processes using spectral methods. In *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, June 2006.

Alan Fern, Sungwook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

Fernando Fernandez and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*, 2006.

Norm Ferns, Pablo Castro, Prakash Panangaden, and Doina Precup. Methods for computing state similarity in Markov decision processes. In *Proceedings of the 22nd Conference on Uncertainty in Artificial intelligence*, pages 174–181, 2006.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for Markov decision processes with infinite state spaces. In *Proceedings of the 2005 Conference on Uncertainty in Artificial Intelligence*, pages 201–208, 2005.

David Foster and Peter Dayan. Structure in the space of value functions. *Machine Learning*, 49 (1/2):325–346, 2004.

Allen Ginsberg. Theory revision via prior operationalization. In *Proceedings of the 1988 National Conference on Artificial Intelligence*, pages 590–595, 1988.

Carlos Guestrin, Daphne Koller, Chris Gearhart, and Neal Kanodia. Generalizing plans to new environments in relational MDPs. In *International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico, August 2003.

Okhtay Ilghami, Hector Munoz-Avila, Dana S. Nau, and David W. Aha. Learning approximate preconditions for methods in hierarchical plans. In *ICML '05: Proceedings of the 22nd International Conference on Machine learning*, pages 337–344, 2005.

Nicholas K. Jong and Peter Stone. Model-based exploration in continuous state spaces. In *The Seventh Symposium on Abstraction, Reformulation, and Approximation*, July 2007.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

Zsolt Kalmár and Csaba Szepesvári. An evaluation criterion for macro learning and some results. Technical Report TR-99-01, Mindmaker Ltd., 1999.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. In *Proc. 15th International Conf. on Machine Learning*, pages 260–268. Morgan Kaufmann, San Francisco, CA, 1998.

W. Bradley Knox and Peter Stone. TAMER: training an agent manually via evaluative reinforcement. In *IEEE 7th International Conference on Development and Learning*, August 2008.

J. Zico Kolter, Pieter Abbeel, and Andrew Ng. Hierarchical apprenticeship learning with application to quadruped locomotion. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 769–776. MIT Press, Cambridge, MA, 2008.

George Konidaris and Andrew Barto. Autonomous shaping: knowledge transfer in reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 489–496, 2006.

George Konidaris and Andrew G. Barto. Building portable options: skill transfer in reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 895–900, 2007.

Gregory Kuhlmann and Peter Stone. Graph-based domain mapping for transfer learning in general games. In *Proceedings of The Eighteenth European Conference on Machine Learning*, September 2007.

Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

John E. Laird, Paul S. Rosenbloom, and Allen Newell. Chunking in soar: the anatomy of a general learning mechanism. *Machine Learning*, 1(1):11–46, 1986.

Alessandro Lazaric. *Knowledge Transfer in Reinforcement Learning*. PhD thesis, Politecnico di Milano, 2008.

Bethany R. Leffler, Michael L. Littman, and Timothy Edmunds. Efficient reinforcement learning with relocatable action models. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 572–577, 2007.

Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for MDPs. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pages 531–539, 2006.

Yaxin Liu and Peter Stone. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 415–20, July 2006.

Richard Maclin and Jude W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22(1-3):251–281, 1996.

Richard Maclin, Jude Shavlik, Lisa Torrey, Trevor Walker, and Edward Wild. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *Proceedings of the 20th National Conference on Artificial Intelligence*, 2005.

Michael G. Madden and Tom Howley. Transfer of experience between reinforcement learning environments with progressive difficulty. *Artificial Intelligence Review*, 21(3-4):375–398, 2004.

Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 8:2169–2231, 2007.

Maja J. Mataric. Reward functions for accelerated learning. In *International Conference on Machine Learning*, pages 181–189, 1994.

John McCarthy. A tough nut for proof procedures. Technical Report Sail AI Memo 16, Computer Science Department, Stanford University, 1964.

Neville Mehta, Sriraam Natarajan, Prasad Tadepalli, and Alan Fern. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73(3):289–312, 2008.

Lilyana Mihalkova and Raymond J. Mooney. Transfer learning by mapping with minimal target data. In *Proceedings of the AAAI-08 Workshop on Transfer Learning for Complex Tasks*, July 2008.

Robin Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

Andrew Moore. Variable resolution dynamic programming: efficiently learning action maps in multivariate real-valued state-spaces. In *Machine Learning: Proceedings of the Eighth International Conference*, June 1991.

Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, October 1993.

Andrew Y. Ng and Michael Jordan. PEGASUS: a policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, 1999.

Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Inverted autonomous helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*, 2004.

Dirk Ormoneit and Saunak Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2-3): 161–178, 2002.

Theodore J. Perkins and Doina Precup. Using options for knowledge transfer in reinforcement learning. Technical Report UM-CS-1999-034, The University of Massachusetts at Amherst, 1999.

Caitlin Phillips. Knowledge transfer in Markov decision processes. Technical report, McGill University, School of Computer Science, 2006. URL http://www.cs.mcgill.ca/~cphill/CDMP/summary.pdf.

Bob Price and Craig Boutilier. Accelerating reinforcement learning through implicit imitation. *Journal of Artificial Intelligence Research*, 19:569–629, 2003.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

Jan Ramon, Kurt Driessens, and Tom Croonenborghs. Transfer learning in reinforcement learning problems through partial policy recycling. In *Proceedings of The Eighteenth European Conference on Machine Learning*, September 2007.

Balaraman Ravindran and Andrew G. Barto. Model minimization in hierarchical reinforcement learning. In *Proceedings of the Fifth Symposium on Abstraction, Reformulation and Approximation*, 2002.

Balaraman Ravindran and Andrew G. Barto. An algebraic approach to abstraction in reinforcement learning. In *Proceedings of the Twelfth Yale Workshop on Adaptive and Learning Systems*, pages 109–114, 2003a.

Balaraman Ravindran and Andrew G. Barto. Relativized options: choosing the right transformation. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pages 608–615, Menlo Park, CA, August 2003b. AAAI Press.

Daniel M. Roy and Leslie P. Kaelbling. Efficient Bayesian task-level transfer learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, Hyderabad, India*, 2007.

Gavin Rummery and Mahesan Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG-RT 116, Engineering Department, Cambridge University, 1994.

Manish Saggar, Thomas D'Silva, Nate Kohl, and Peter Stone. Autonomous learning of stable quadruped locomotion. In Gerhard Lakemeyer, Elizabeth Sklar, Domenico Sorenti, and Tomoichi Takahashi, editors, *RoboCup-2006: Robot Soccer World Cup X*, volume 4434 of *Lecture Notes in Artificial Intelligence*, pages 98–109. Springer Verlag, Berlin, 2007.

Oliver G. Selfridge, Richard S. Sutton, and Andrew G. Barto. Training and tracking in robotics. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 670–672, 1985.

Lloyd S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39(10):1095–1100, October 1953.

Manu Sharma, Michael Holmes, Juan Santamaria, Arya Irani, Charles Isbell, and Ashwin Ram. Transfer learning in real-time strategy games using hybrid CBR/RL. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.

Alexander A. Sherstov and Peter Stone. Improving action selection in MDP's via knowledge transfer. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, July 2005.

Danny Silver, Goekhan Bakir, Kristin Bennett, Rich Caruana, Massimiliano Pontil, Stuart Russell, and Prasad Tadepalli. NIPS workshop on "Inductive transfer: 10 years later", December 2005.

Özgür Şimşek and Andrew G. Barto. An intrinsic reward mechanism for efficient exploration. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, 2006.

Satinder Singh and Richard S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158, 1996.

Satinder P. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339, 1992.

Burrhus F. Skinner. *Science and Human Behavior*. Colliler-Macmillian, 1953.

Vishal Soni and Satinder Singh. Using homomorphisms to transfer options across continuous reinforcement learning domains. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*, July 2006.

Ashwin Srinivasan. The aleph manual, 2001.

Peter Stone and Manuela Veloso. Multiagent systems: a survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.

Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

Funlade T. Sunmola and Jeremy L. Wyatt. Model transfer for Markov decision tasks via parameter matching. In *Proceedings of the 25th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG 2006)*, December 2006.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

Richard S. Sutton, Anna Koop, and David Silver. On the role of tracking in stationary environments. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

Samarth Swarup and Sylvian R. Ray. Cross-domain knowledge transfer using structured representations. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*, July 2006.

Umar Syed and Robert Schapier. A multiplicative weights algorithm for apprenticeship learning. In *Advances in Neural Information Processing Systems 21*, 2007.

Erik Talvitie and Satinder Singh. An experts algorithm for transfer learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.

Fumihide Tanaka and Masayuki Yamamura. Multitask reinforcement learning on the distribution of MDPs. *Transactions of the Institute of Electrical Engineers of Japan. C*, 123(5):1004–1011, 2003.

Brian Tanner, Adam White, and Richard S. Sutton. RL Glue and codecs, 2008. `http://mloss.org/software/view/151/`.

Matthew E. Taylor and Peter Stone. Representation transfer for reinforcement learning. In *AAAI 2007 Fall Symposium on Computational Approaches to Representation Change during Learning and Development*, November 2007a.

Matthew E. Taylor and Peter Stone. Cross-domain transfer for reinforcement learning. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, June 2007b.

Matthew E. Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125–2167, 2007a.

Matthew E. Taylor, Shimon Whiteson, and Peter Stone. Transfer via inter-task mappings in policy search reinforcement learning. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2007b.

Matthew E. Taylor, Alan Fern, Kurt Driessens, Peter Stone, Richard Maclin, and Jude Shavlik. AAAI workshop on "Transfer learning for complex tasks", July 2008a.

Matthew E. Taylor, Nicholas Jong, and Peter Stone. Transferring instances for model-based reinforcement learning. In *Proceedings of the Adaptive Learning Agents and Multi-Agent Systems (ALAMAS+ALAG) workshop at AAMAS-08*, May 2008b.

Matthew E. Taylor, Nicholas K. Jong, and Peter Stone. Transferring instances for model-based reinforcement learning. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 488–505, September 2008c.

Gerald Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

Edward L. Thorndike and Robert S. Woodworth. The influence of improvement in one mental function upon the efficiency of other functions. *Psychological Review*, 8:247–261, 1901.

Sebastian Thrun. Is learning the *n*-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, volume 8, pages 640–646, 1996.

Sebastian Thrun and Lorien Pratt, editors. *Learning to learn*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

Lisa Torrey, Trevor Walker, Jude W. Shavlik, and Richard Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 412–424, 2005.

Lisa Torrey, Jude W. Shavlik, Trevor Walker, and Richard Maclin. Skill acquisition via transfer learning and advice taking. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 425–436, 2006.

Lisa Torrey, Jude W. Shavlik, Trevor Walker, and Richard Maclin. Relational macros for transfer in reinforcement learning. In *Proceedings of the Seventeenth Conference on Inductive Logic Programming*, 2007.

Thomas J. Walsh, Lihong Li, and Michael L. Littman. Transferring state abstractions between MDPs. In *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, June 2006.

Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

Shimon Whiteson, Adam White, Brian Tanner, Richard S. Sutton Sutton, Doina Precup, Peter Stone, Michael Littman, Nikos Vlassis, and Martin Riedmiller. ICML workshop on "The 2008 RL-competition", July 2008.

Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical Bayesian approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 1015–1022, 2007.

Wei Zhang and Thomas G. Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

Journal of Machine Learning Research 10 (2009) 1687-1704 Submitted 3/08; Revised 2/09; Published 7/09

Application of Non Parametric Empirical Bayes Estimation to High Dimensional Classification

Eitan Greenshtein EITAN.GREENSHTEIN@GMAIL.COM
Central Bureau of Statistics
66 Kanfei Nesharim St
Jerusalem, 95464, Israel

Junyong Park JUNPARK@UMBC.EDU
Department of Mathematics and Statistics
University of Maryland Baltimore County
Baltimore, MD 21250, USA

Editor: Bin Yu

Abstract

We consider the problem of classification using high dimensional features' space. In a paper by Bickel and Levina (2004), it is recommended to use naive-Bayes classifiers, that is, to treat the features as if they are statistically independent.

Consider now a sparse setup, where only a few of the features are informative for classification. Fan and Fan (2008), suggested a variable selection and classification method, called FAIR. The FAIR method improves the design of naive-Bayes classifiers in sparse setups. The improvement is due to reducing the noise in estimating the features' means. This reduction is since that only the means of a few selected variables should be estimated.

We also consider the design of naive Bayes classifiers. We show that a good alternative to variable selection is estimation of the means through a certain non parametric empirical Bayes procedure. In sparse setups the empirical Bayes implicitly performs an efficient variable selection. It also adapts very well to non sparse setups, and has the advantage of making use of the information from many "weakly informative" variables, which variable selection type of classification procedures give up on using.

We compare our method with FAIR and other classification methods in simulation for sparse and non sparse setups, and in real data examples involving classification of normal versus malignant tissues based on microarray data.

Keywords: non parametric empirical Bayes, high dimension, classification

1. Introduction

We consider the problem of finding a classifier for a response variable $Y \in \{-1,1\}$ based on a vector $(X_1,...,X_p) \in \mathbb{R}^p$ of explanatory variables.

Suppose we have a 'training set' (or a sample) of n_1 examples $(Y_i,X_{i1},...,X_{ip})$, $i = 1,...,n_1$, for which $Y_i = -1$, and additional n_2 examples $(Y_i,X_{i1},...,X_{ip})$, $i = n_1+1,...,n_1+n_2$, for which $Y_i = 1$. We assume that the n_1+n_2 observations are independent. In what follows we assume for simplicity that $n_1 = n_2 \equiv n$.

Our study is aimed to understand and suggest a good classification procedure in a high dimensional setup. Here, by high dimensionality we mean $p \gg n$. There are many examples in

contemporary statistical applications where $p \gg n$. We mention that of microarray data where the dimensionality is typically of thousands, while the sample size is of the order of dozens or hundreds.

In particular we focus on linear predictors for Y, which are of the form:

$$\hat{Y} = \text{sign}\left(\sum_{j=1}^{p} a_j X_j + a_0\right),$$

where $a_0, a_1, ..., a_p$ are constants.

Suppose the distribution of the explanatory variables, conditional on $Y = -1$ and on $Y = 1$, is G_1 and G_2 correspondingly, where G_i are multivariate normals $i = 1, 2$. Assume that the covariance matrices of G_i, $i = 1, 2$ are the same. Then the optimal classifier is Fisher's rule. However when the common covariance matrix as well as the vectors of means under G_1 and G_2 are unknown, we can not apply Fisher's rule. When $n \gg p$, the naive approach, of estimating the unknown quantities and plug-in to Fisher's rule, would work well. It is impractical when $p \gg n$. A practical solution, called 'naive Bayes' is to neglect estimation of the off diagonal elements in the covariance matrix (or to estimate them trivially) by setting those values to be 0. Then apply Fisher's rule by plugging in the estimated diagonal covariance matrix and the estimated vectors of means. Bickel and Levina (2004) showed that in many cases, by this trivial estimation of the covariance matrix, one does not lose too much in terms of classification error, relative to incorporating the true covariance matrix, and suggested this practice. Note, the bottom line of this practice is to treat the explanatory variables as if they are independent, or act "assuming" independence of the explanatory variables. We will also refer in the sequel to Fisher's rule as the Independence Rule, or IR.

It was pointed out independently by Fan and Fan (2008) and by Greenshtein et al. (2009), that even in the independent case, when $p \gg n$, estimating the vector of means under G_1 and under G_2 by the corresponding sample averages, could lead to a very weak estimator, resulting in a corresponding classifier with virtually no classification power (see Theorem 1 in Fan and Fan 2008, and Proposition 1 of Greenshtein et al. 2009). This is also in cases where there exists a good linear classifier. In other words, often, attempting to estimate the $2p$ coordinates of the two mean vectors, by the corresponding averages of n observations on each, is already "too much", and leads to overfit. The FAIR approach suggested by Fan and Fan (2008), and the conditional MLE approach suggested by Greenshtein et al. (2009), are based on variable selection techniques followed by estimation of the mean of the selected explanatory variables, while ignoring the others (i.e., setting the corresponding coefficients of the linear classifier to be equal to zero). The FAIR method estimates the means of the selected variables by the corresponding sample means (the MLE), while the conditional MLE method estimates by the conditional MLE, conditional on the event that the variables were selected.

The above approaches are helpful especially in a high dimensional sparse setup, while the non parametric Empirical Bayes approach that we will present is helpful also in non-sparse setups. Let μ and τ be the vectors of means under G_1 and G_2 correspondingly; here 'sparse' setup means that the vector

$$\nu \equiv \mu - \tau$$

is sparse. A 'sparse' setup is such, that relatively few of the explanatory variables are informative for the classification.

1.1 On Types of Sparsity

The term sparse vector is only loosely defined in the literature, and we will keep some of the ambiguity. However, by a sparse vector ν we mean that most of its coordinates are *exactly* zero. Throughout our study we consider only vectors ν such that their l_2 norm $||\nu||$, is much smaller than their dimension, say, $||\nu|| = o(p)$. The last condition *does not* imply sparsity under our terminology.

We concentrate on configurations such that $||\nu|| = o(p)$, since that as $p \to \infty$, when letting $||\nu|| = O(p)$, any reasonable procedure would achieve asymptotically (virtually) zero misclassification rate. We are interested in the cases when there is not enough signal to make nearly perfect classification, that is, $p \gg ||\nu||$. In our simulation, we achieve $p \gg ||\nu||$, by considering the following three types of configurations for vectors $||\nu||$:

(a) Very few non-zero coordinates of a large/moderate magnitude (i.e., sparse vectors)

(b) Very few coordinates of a large magnitude, mixed with many very small coordinates (i.e., non-sparse vectors).

(c) Many coordinates of a very small magnitude (i.e., non-sparse vectors).

In sparse configurations, our EB procedure is comparable to the other procedures. Specifically, it is better in moderately sparse setups, while in extremely sparse cases, it is inferior. Indeed, when there are only a few relevant variables, naturally methods which are based on variable selection would do well. In non-sparse configurations our EB procedure is clearly advantageous in simulations. This is in line with the theoretical results in Brown and Greenshtein (2009), and in Jiang and Zhang (2007), on optimality of non-parametric empirical Bayes in estimation of high dimensional not extremely sparse normal mean vectors, coupled with the relation between estimation and classification as explained in Section 2.

The above mentioned results, join a huge body of literature on Empirical Bayes starting with Robbins (1951), see the surveys by Copas (1969) and by Zhang (2003). See also a recent paper by Greenshtein and Rotov (2009) on efficiency of compound and empirical Bayes procedures with respect to the class of permutation invariant procedures. A recent comprehensive study and performance comparison, of various methods for estimating a vector of normal means under squared error loss, was conducted by Brown (2008), the very good performance of non parametric empirical Bayes methods is demonstrated also there. Our approach is related to (and independent of) the approach in Efron (2009), where EB estimation method is used to obtain good classifiers.

We will introduce and explain the virtues of our empirical Bayes classification method and provide simulation evidence as well as real data evidence to its excellent performance. We will compare the performance of our Empirical Bayes classifiers to that of FAIR (Fan and Fan, 2008), conditional MLE (Greenshtein et al. , 2009), NSC (Tibshirani et al. , 2002), and plug in Fisher's rule.

The outline is the following. In the next section we introduce our formal setup and explain the relation between estimating a vector of means under a squared loss and classification. In Section 3 we introduce a class of non-parametric empirical Bayes estimators of a vector of normal means and define our classifier. In Section 4 we demonstrate the performance of our classifier on simulated as well as real data.

2. Preliminaries

Assume a multivariate normal distribution of the vector $(X_1,...,X_p)$ conditional on the value of Y. Specifically, we assume $(X_j|Y=-1) \sim N(\mu_j, s^2)$ and $(X_j|Y=1) \sim N(\tau_j, s^2)$ independently, $j=1,...,p$. We will assume that the variance s^2 is known. Denote $\mu=(\mu_1,...,\mu_p), \tau=(\tau_1,...,\tau_p)$.

In considering linear classifiers, when both p and n are large it is robust to assume normality of $(X_1,...,X_p)$ by the central limit theorem. Due to Lindberg's CLT, large p implies that $\sum a_j X_j$ will be close to normal, when a_j are comparable in size, even if the individual X_j are not normal. In addition, large n implies that averages of independent X_{ij}, $i=1,...,n$ (as in Z_j, which is defined in the sequel) are close to normal. The CLT arguments are problematic when the X_js have heavy tails. In Table 5 of Section 4 some simulations are carried to demonstrate the effect of heavy tailed distributions.

When searching for values $a \equiv (a_1,...,a_p)$ that determine a 'good' linear classifier, we assume w.l.o.g. that $\|a\|^2 = \sum_{j=1}^p a_j^2 = 1$. In this case the optimal choice of $(a_1,...,a_p)$ is the vector that maximizes $|\sum a_j\mu_j - \sum a_j\tau_j|$. Note that the optimal choice of $a_1,...,a_p$ is the same regardless of the misclassification loss (the value of a_0 does depend on the loss). In order to see it, observe that $\sum a_j X_j \sim N(\sum a_j\mu_j, s^2) \equiv N(\theta_1, s^2)$ conditional on $Y=-1$ and $\sum a_j X_j \sim N(\sum a_j\tau_j, s^2) \equiv N(\theta_2, s^2)$ conditional on $Y=1$; here $\theta_i, = 1,2$ are implicitly defined. Hence, an optimal choice of $a_1,...a_p$ is such that

$$V = V(a_1,...,a_n) \equiv |\sum_j a_j\mu_j - \sum_j a_j\tau_j| = |\theta_1 - \theta_2| \tag{1}$$

is maximized. This implies that the coordinates a_j^{opt} of the optimal choice satisfy:

$$a_j^{opt} = \frac{\nu_j}{\sqrt{\sum \nu_j^2}},\ j=1,...p; \tag{2}$$

recall $\nu_j = \mu_j - \tau_j$.

Under a 0-1 loss, given any choice of $(a_1,...,a_p)$, the corresponding minmax choice of a_0 is

$$a_0 = -\frac{\theta_2+\theta_1}{2}.$$

This is also the Bayes solution assuming a prior $\pi_i = 0.5$ for each class. The optimal choice of a_0 for none-equal losses and priors is straightforward.

A formal argument showing that the optimal $a_1,...,a_p$ is the same regardless of the misclassification loss may be obtained using the theory of comparison of experiments, implying that the experiment that consists of the distributions $N(\theta_1, s^2)$ and $N(\theta_2, s^2)$, dominates the experiment that consists of the distributions $N(\theta_1', s^2)$ and $N(\theta_2', s^2)$ if and only if $|\theta_1 - \theta_2| \geq |\theta_1' - \theta_2'|$. See Lehmann (1986, p. 86), for some basic theory on comparison of experiments and some additional references.

By the above discussion there is a natural order relation $\preceq$ between two classifiers determined by a and a'. We say that $a \preceq a'$ if for the corresponding θ_i and θ_i',

$$V = |\theta_1' - \theta_2'| \geq |\theta_1 - \theta_2| = V' \tag{3}$$

Note, here $V \equiv V(a_1,...,a_p)$, is a function of $(a_1,...,a_p)$.

By (2), $V(a_1^{opt},...,a_p^{opt}) = ||\nu||$, consequently for the optimal choice a_0^{opt}, the Bayes risk is:

$$\Phi(-\frac{||\nu||}{2s}) \tag{4}$$

where Φ is the cumulative distribution of a standard normal distribution.

2.1 Summary

The goal of finding the optimal classifier when $\nu_j,\ j=1,...,p$ are unknown, is not practical. However we want to find a classifier with a corresponding 'large' value of V.

Note, in statistical inference the choice of $a_j,\ j=1,...,p$ depends on the data. The dependence on the data is through the vector

$Z=(Z_1,...,Z_p)$; here, for $n=n_1=n_2$

$$Z_j=\frac{\sum_{i=1}^{n}X_{ij}}{n}-\frac{\sum_{i=n+1}^{2n}X_{ij}}{n},\ j=1,...,p, \tag{5}$$

are independent normal random variables with $EZ_j=\nu_j$ and variance, denoted σ^2,

$$\sigma^2=\frac{2s^2}{n}. \tag{6}$$

Thus, depending on the particular procedure the selected value of a_j depends on $Z_1,...,Z_p$, and it is a random variable denoted $\hat{a}_j,\ j=1,...,p$.

Equation (3), motivates us to search for procedures with high value of

$$E(V)=E|\sum_{j=1}^{p}\hat{a}_j\nu_j|.$$

Thus we extend the definition of the order relation, to apply to two statistical procedures $\{\hat{a}_j\},\ j=1,...,p$, and $\{\hat{a}'_j\},\ j=1,...,p$.

Definition 1: We say that $\{\hat{a}'_j\},\ j=1,...,p$, dominates $\{\hat{a}_j\},\ j=1,...,p$, if for the corresponding V' and V, $E(V')\geq E(V)$.

Remark 1: Evaluating a procedure $\hat{a}_j\ j=1,...,p$, by its corresponding value $E(V)$, is simplistic, for example, it ignores the effect of the standard deviation of V on the classification error. However, in high dimensional setup one might hope that the standard deviation of V is small compare to $E(V)$. Otherwise, one might perceive it as a convenient approximate evaluation. Note however, that for two procedures with very accurate classification rate, ignoring the variability of V might be significantly misleading even if $E(V)$ is large compare to the standard deviation of V, this is due to the thin tail of the normal distribution.

2.2 On the Relation Between Estimating the Mean Under a Squared Loss and Classification

Since the optimal choice of $a_j,\ j=1,...,p$, is $a_j^{opt}=\frac{\nu_j}{\sqrt{\sum\nu_j^2}}$, a natural way to proceed is to estimate ν_j by a 'good' estimator $\hat{\nu}_j$ for ν_j, and then plug-in, that is, let $\hat{a}_j=\frac{\hat{\nu}_j}{\sqrt{\sum\hat{\nu}_j^2}}$. A formal definition of 'good' in the above, depends on the loss function. In the sequel we will indicate why the squared error loss function is especially appropriate.

First we state the obvious. In general, the fact that $\hat{\nu}$ is a good estimator for ν under a squared error loss, does not indicate that $T(\hat{\nu})$ is a good estimator for $T(\nu)$ under (say) a squared loss. For example in the case $T(\nu)=\sum\nu_j$, plugging in the MLE for ν will often be better than plugging in the James-Stein estimator because of the bias of the J-S estimator which is accumulated. This is although the J-S estimator dominates the MLE in estimating ν under a squared error loss. Hence good properties of the Empirical Bayes as an estimator for ν under squared loss in high dimensions,

do not automatically indicate that it should be plugged-in in order to obtain good estimators for a_j^{opt}, and thus provide good classifiers.

Consider the collection of all vectors $(a_1,...,a_p)$ with l_2 norm 1. Define the function

$$L((a_1,...,a_p)) = \sum(\nu_j - a_j)^2$$

Then, one may check that on the surface of the p dimensional unit ball,

$$L(a) = -2 \times V(a) + C,$$

where $C = 1 + \sum \nu_j^2$, and V is defined in (1).

The last equation motivates the particular choice of a squared error loss when evaluating an estimator $\hat{\nu}_j$. This is because of the direct relation between minimizing $E(L)$ to that of maximizing $E(V)$. Maximizing V is crucial in obtaining a good classifier, as explained in the first part of this section.

An estimator with particularly appealing properties, in estimation of a vector of means under a squared loss in high dimensions, is the non-parametric empirical Bayes estimator, see Brown and Greenshtein (2009). We describe it in the following section and then define our procedure.

For a given ν the success in obtaining a good classifier has to do with two aspects. The larger is the l_2 norm of ν the smaller is the misclassification rate of the Bayes procedure, as may be seen in (4), and typically also the misclassification rate of our EB procedure. The more difficult is the task of estimating ν by our non parametric empirical Bayes method in terms of MSE, the less successful is our classification method. As pointed by a referee, the difficulty/MSE in estimating ν by EB is invariant under translation, while (obviously) the l_2 norm is not. When the vector ν is identically zero (i.e., no signal) the corresponding misclassification rate is 0.5. The corresponding rates for various translations of the zero-vector may be found in Table 4.

3. Empirical Bayes Classification

In this section, we define our linear classifier for the cases of known homoscedastic variances and unknown heteroscedastic variances.

3.1 Known Homoscedastic Variance

In the sequel we rescale X_j, so that Z_j defined in (5) will have variance $\sigma^2 = 1$, $j = 1,...,p$. This is possible since s, the common standard deviation of X_j, is known see (6). When the variances are unknown (and not assumed equal) we simply standardize the variables using the sample variance. The extension of this subsection for the latter case and for non equal samples n_1 and n_2 is explicitly given in the next subsection.

Under the non-parametric empirical Bayes approach for estimating a vector of means, we consider the means $\nu_i = E(Z_i)$, $i = 1,...,p$, as realizations of i.i.d random variables $M_1,...,M_p$ distributed G, where G is completely unknown. Still, we attempt to approximate the Bayes estimator of the mean, denoted $\delta^G(z)$, by $\hat{\delta}(z)$. Then we estimate ν_i by $\hat{\nu}_i = \hat{\delta}(Z_i)$.

More formally it is described in the following. Let $Z \sim N(M,1)$ where $M \sim G, G \in \mathcal{G}$. We want to emulate the Bayes procedure δ^G based on a sample $Z_1,...,Z_p$, $Z_i \sim N(M_i,1)$, $i = 1,...,p$, where $M_i \sim G$ and the Z_i are independent conditional on $M_1,...,M_p$, $i = 1,...,p$.

Let g^* be the mixture density

$$g^*(z) = \int \phi(z-\nu) dG(\nu).$$

Then from Brown (1971) equation (1.2.2), we have that the Bayes procedure, denoted δ^G, satisfies

$$\delta^G(z) = z + \frac{g^{*'}(z)}{g^*(z)};$$

here $g^{*'}(z)$ is the derivative of $g^*(z)$. The estimator that we suggest for δ^G, is of the form

$$\hat{\delta}_h(z) = z + \frac{\hat{g}_h^{*'}(z)}{\hat{g}_h^*(z)}$$

where $\hat{g}_h^{*'}(z)$ and $\hat{g}_h^*(z)$ are appropriate kernel estimators for the density $g^*(z)$ and its derivative $g^{*'}(z)$. The subscript h denotes the bandwidth of the kernel estimator. We will use a normal kernel.

Let $h > 0$ be a bandwidth constant. Then define the kernel estimator

$$\hat{g}_h^*(z) = \frac{1}{nh} \sum \phi(\frac{z - Z_i}{h}).$$

Its derivative is:

$$\hat{g}_h^{*'}(z) = \frac{1}{nh} \sum \frac{Z_i - z}{h} \times \phi(\frac{z - Z_i}{h}).$$

In Brown and Greenshtein (2009), it is suggested to let the bandwidth converge slowly to zero as $p \to \infty$, they suggested that h^2 should approach zero 'just faster' than $1/\log(p)$. In the simulations and real data analysis in this paper, we applied $h = 0.3 \approx 1/\sqrt{\log(p)}$, which is in agreement with that suggestion for the range of features' dimensions p that we study. The choice $h = 1/\sqrt{\log(p)}$ is also suggested in Brown and Greenshtein (2009) as a 'default' choice. A more careful choice could involve, for example, cross validation. However, the results are not too sensitive to the choice.

3.2 The Empirical Bayes Classifier

We now define our Empirical Bayes classifier.

Let

$$\hat{\nu}_i = \hat{\delta}_h(Z_i), \;\; i = 1, ..., p.$$

Let

$$\hat{a}_i = \frac{\hat{\nu}_i}{\sqrt{\sum_j \hat{\nu}_j^2}} \;\; i = 1, ..., p.$$

In order to fully define our classifier, we should still define the parameter $\hat{a}_0$, given $\hat{a}_1, ..., \hat{a}_p$. We do it for the case of 0-1 loss and equal prior probabilities for each class. An obvious way is the following. Let $\hat{\theta}_1 = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^p \hat{a}_j X_{ij}$, where the summation is over the n examples $(Y_i, X_{i1}, ..., X_{ip})$ for which $Y_i = -1$. Similarly define $\hat{\theta}_2$.

Let,

$$\hat{a}_0 = -\frac{\hat{\theta}_2 + \hat{\theta}_1}{2}.$$

where we assume w.l.o.g. that $\hat{\theta}_1 < \hat{\theta}_2$.

3.3 Unknown Heteroscedastic Variances

Consider now the case where the standard deviation, denoted s_j, of X_j are unknown $j = 1,...,p$. We now introduce a superscript $k = 1,2$ to denote quantities associated with the data corresponding to $Y = -1$ and $Y = 1$. Denote by $\hat{s}_j^k$ the usual estimates of the standard deviation of X_j^k. The estimates are based on the corresponding X_{ij}^k, $k = 1,2$, $i = 1,...,n_k$, $j = 1,...,p$. Denote $\bar{X}_j^1$ and $\bar{X}_j^2$ the corresponding means.

Let

$$\hat{S}_j = \sqrt{\frac{(\hat{s}_j^1)^2}{n_1} + \frac{(\hat{s}_j^2)^2}{n_2}},$$

thus $\hat{S}_j$ is our estimator for the standard deviation of $\bar{X}_j^1 - \bar{X}_j^2$.

Let

$$Z_j = \frac{\bar{X}_j^1 - \bar{X}_j^2}{\hat{S}_j};$$

note, we expect that the variance of Z_j is approximately 1, $j = 1,...,p$.

As before let $\hat{\nu}_i$ be the empirical Bayes estimators of $E(Z_i)$, and let $\hat{a}_i = \frac{\hat{\nu}_i}{\sqrt{\sum_j \hat{\nu}_j^2}}$, $i = 1,...,p$.

In the following we proceed in terms of the variables

$$U_j = \frac{X_j}{\hat{S}_j} \quad j = 1,...,p.$$

We will represent our linear classifiers as linear functions of U_j, $j = 1,...,p$.

Let

$$\hat{\theta}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \sum_{j=1}^{p} \hat{a}_j U_{ij}^k, \; k = 1,2.$$

Let

$$\hat{a}_0 = -\frac{\hat{\theta}_2 + \hat{\theta}_1}{2}.$$

Finally, our classifier is:

$$\text{sign}(\sum_{j=1}^{p} \hat{a}_j U_j + \hat{a}_0).$$

4. Simulations and Data Analysis

In this section, we present numerical studies including simulations and application to three sets of real data.

4.1 Simulations

The simulation study in this subsection is based on the procedure described in Section 3.1. We present simulations for $p = 10^5$ and for $p = 10^4$, under various configurations in which $\tau_j = 0$, $j = 1,...,p$. We study sparse configurations where for l variables the corresponding mean is fixed $\nu_j = \Delta$, while the remaining $p - l$ variables have $\nu_j = 0$, $p \gg l$. We also study a non-sparse version of the above where the remaining $p - l$ variables have means ν_j which are randomly selected from

$N(0,0.1^2)$. The small variance of the normal distribution is in order to control the magnitude of $||\nu||$; recall from the introduction, we want $p \gg ||\nu||$. Thus a configuration is determined by (Δ, l), the corresponding p, and whether the $p-l$ coordinates, whose means are not equal to Δ, are set to be equal to zero or, alternatively get their values randomly based on a $N(0,0.1^2)$ distribution.

We consider the case where $n = 25$, and a rescale under which the variance of Z_j is $\sigma^2 = 1$, $j = 1,...,p$. Thus, the variance of X_j is $s^2 = 25/2$, and the same for the variance of $\sum a_j X_j$, when $\sum a_j^2 = 1$. The distribution X_j is normal throughout this section, except for the simulations reported in Table 5. In Table 5 the effect of a heavy tailed distribution variables X_j is studied.

Table 1 shows the misclassification rates of the empirical Bayes, conditional MLE, FAIR, and the plug in Fisher's rule which is also termed IR (Independence Rule). The plug in Fisher's refers to plugging in Z_j for ν_j, $j = 1,...,p$, in Fisher's rule. We see that the empirical Bayes approach produces the best results for non-sparse and for moderately sparse configurations. The CMLE is better for strongly sparse configurations. The version of FAIR we are using is described in Theorem 4 of Fan and Fan (2008). It performs similar to IR, since it selects too many variables. Fan and Fan (2008) describe another version of FAIR in their equation (4.3), this other version screens variables more aggressively and involves computation of eigenvalues of the empirical covariance matrix. That more aggressive version might perform better in our simulation, yet it is motivated for cases where it is not known that the covariance matrix is of the form $\sigma^2 I$ (which is used in most of our simulations). In addition, computing eigenvalues for empirical covariance matrix with $p = 10^5$ is computationally intensive. In the real data analysis, with unknown covariance matrix, the other version of FAIR is used.

Each entry is based on simulated $Z_1,...,Z_p$, and on calculating the exact theoretical misclassification rate. Note, given the estimators $\hat{a}_j$ $j = 0,1,...,p$, for a given simulated realization, the theoretical misclassification error, under equal prior probability for each class, is $\frac{1}{2}\Phi((-\sum_{j=1}^{p}\hat{a}_j\mu_j - \hat{a}_0)/s) + \frac{1}{2}(1-\Phi((-\sum_{j=1}^{p}\hat{a}_j\tau_j - \hat{a}_0))/s)$.

In order to demonstrate the effect of dependence and to compare the methods for correlated variables, we also consider correlated normal variables where the correlation of X_i and X_j, namely ρ_{ij}, has the form of $\rho^{|i-j|}$ known as AR(1) model. Here, the corresponding misclassification probabilities are $\frac{1}{2}\Phi((-\sum_{j=1}^{p}\hat{a}_j\mu_j - \hat{a}_0)/\sqrt{\hat{\mathbf{a}}'S\hat{\mathbf{a}}}) + \frac{1}{2}(1-\Phi((-\sum_{j=1}^{p}\hat{a}_j\tau_j - \hat{a}_0))/\sqrt{\hat{\mathbf{a}}'S\hat{\mathbf{a}}})$, for the appropriate covariance matrix S.

Table 2 presents misclassification rates under different values of ρ. The empirical Bayes still achieves the lowest error rates for all those non-sparse configurations. The reported entries are averages of the 100 error rates corresponding to 100 realizations and corresponding estimators $\hat{a}_j$ for the particular configuration (Δ, l) or $(\Delta_1, l_1, \Delta_2, l_2)$ where $(\Delta_1, l_1, \Delta_2, l_2)$ means that l_1 and l_2 coordinates in ν are all valued Δ_1 and Δ_2 correspondingly, while the remaining entries are all zero.

In Table 3 we present simulation results under the following correlation structure which is much heavier than that of AR(1). We consider correlations $corr(X_i, X_j) = \rho_{ij} = \alpha_i\alpha_j$ for $i \neq j$ which is easily implemented by letting $X_i = \tau_i(\text{or } \mu_i) + \sqrt{1-\alpha_i^2}W_i + \alpha_i U$ where W_i's $1 \leq i \leq p$ and U are generated independently from $N(0,s^2)$. In our simulations, all $\alpha_i's$ are generated from $U(-a,a)$ where $a = 0.3, 0.5, 0.7$ and 0.9 are considered. As a increases, variables are more correlated. Table 3 shows misclassification probabilities for configurations of (Δ, l) or $(\Delta_1, l_1, \Delta_2, l_2)$ as in Table 2.

In general the effect of correlation (especially heavy positive correlation) on the EB classification method is stronger than on the other methods. This is partially because the EB uses more

	$p = 10^4$							
	$p-l$ Δ's are 0				$p-l$ Δ's $\sim N(0,0.1^2)$			
(Δ, l)	EB	CMLE	FAIR	IR	EB	CMLE	FAIR	IR
(1.0, 2000)	*0.0003	0.0091	0.0052	0.0052	*0.0000	0.0082	0.0049	0.0049
(1.0, 1000)	*0.0396	0.1309	0.0906	0.0905	*0.0162	0.1182	0.0854	0.0852
(1.0, 500)	*0.2006	0.3243	0.2475	0.2474	*0.1055	0.3139	0.2341	0.2339
(1.5, 300)	*0.1172	0.1891	0.1805	0.1806	*0.0657	0.1771	0.1679	0.1680
(2.0, 200)	*0.0521	0.0868	0.1413	0.1416	*0.0340	0.0813	0.1315	0.1318
(2.5, 100)	*0.0529	0.0631	0.1985	0.1990	*0.0396	0.0632	0.1863	0.1867
(3.0, 50)	0.0641	*0.0604	0.2677	0.2682	*0.0518	0.0583	0.2532	0.2536
(3.5, 50)	0.0113	*0.0099	0.2019	0.2025	*0.0093	0.0095	0.1893	0.1898
(4.0, 40)	0.0042	*0.0033	0.1933	0.1939	0.0039	*0.0034	0.1807	0.1812
	$p = 10^5$							
	$p-l$ Δ's are 0				$p-l$ Δ's $\sim N(0,0.1^2)$			
(Δ, l)	EB	CMLE	FAIR	IR	EB	CMLE	FAIR	IR
(1.0, 2000)	*0.1444	0.2717	0.1896	0.1894	*0.0077	0.2364	0.1542	0.1539
(1.0, 1000)	*0.3100	0.3952	0.3294	0.3293	*0.0367	0.3721	0.2792	0.2789
(1.0, 500)	*0.4067	0.4552	0.4122	0.4121	*0.0702	0.4408	0.3567	0.3565
(1.5, 300)	*0.3643	0.3868	0.3817	0.3818	*0.0628	0.3744	0.3278	0.3278
(2.0, 200)	*0.3071	0.2865	0.3609	0.3611	*0.0522	0.2727	0.3080	0.3081
(2.5, 100)	0.3009	*0.2275	0.3901	0.3903	*0.0560	0.2261	0.3358	0.3359
(3.0, 50)	0.3006	*0.1887	0.4202	0.4204	*0.0597	0.1886	0.3649	0.3649
(3.5, 50)	0.1521	*0.0474	0.3927	0.3929	*0.0263	0.0507	0.3387	0.3388
(4.0, 40)	0.0792	*0.0162	0.3876	0.3879	*0.0121	0.0157	0.3339	0.3340

Table 1: Misclassification error rates by Empirical Bayes, conditional MLE (Greenshtein et al. 2009), FAIR (Fan and Fan 2008) and Fisher's rule (i.e., without variable selection)) . Error rate with * represents minimum error rate in the row for the corresponding configuration.

variables, so more correlations are in effect, relative to variable selection methods that screen variables and consequently their correlations do not effect.

In Table 4, we compare the above mentioned procedures in non sparse setups where there are many small signals. In all the configurations there is 'enough overall signal' to make virtually no classification error if μ and τ were known. In those configurations the optimal (unknown) linear classifiers uses most (or all) of the variables. However, attempting to estimate the corresponding means by FAIR or Fisher's plug-in and the Conditional MLE methods yield poor classifiers, while the non parametric empirical Bayes method yields classifiers with excellent performance in some cases.

In Table 5, we present simulation studies for X_js with a heavy tailed distribution. As before $n_1 = n_2 = 25$. Under G_1 the distribution of X_j is $c \times t(3)$ (i.e., t with 3 degrees of freedom), $j = 1,...,p$ where c is chosen so that the variance of X_j is $s^2 = 25/2$. Under G_2 the distribution of X_j is $\nu_j + c \times X_j$, where X_j is distributed $t(3)$, $j = 1,...,p$. Thus, the corresponding Z_j has variance 1 and it is only approximately normal. We study the configurations $(\Delta, l) = (1,2000), (2.5,100), (3.5,50)$ and $(4,40)$, which were also studied in Table 1. The misclassification rates are obtained based on test sets of size $1000, 500$ from each G_i, $i = 1,2$. As seen in Table 5, the EB method and CMLE still

	$(\Delta, l) = (1, 2000)$, $p - l$ $\Delta \sim N(0, 0.1^2)$							
	$p = 10^4$				$p = 10^5$			
ρ	EB	CMLE	FAIR	IR	EB	CMLE	FAIR	IR
0.3	*0.0014	0.0158	0.0131	0.0119	*0.0216	0.2532	0.1801	0.1777
0.5	*0.0082	0.0355	0.0315	0.0303	*0.0487	0.2723	0.2177	0.2175
0.7	*0.0391	0.0850	0.0798	0.0797	*0.1094	0.3149	0.2765	0.2781
0.9	*0.1679	0.2256	0.2166	0.2176	*0.2508	0.3888	0.3703	0.3721
	$(\Delta_1, l_1, \Delta_2, l_2) = (2.5, 100, 1, 1000)$ and $p - l_1 - l_2, \Delta' s \sim N(0, 0.1^2)$							
	$p = 10^4$				$p = 10^5$			
ρ	EB	CMLE	FAIR	IR	EB	CMLE	FAIR	IR
0.3	*0.0051	0.0261	0.0295	0.0301	*0.0320	0.1904	0.2164	0.2175
0.5	*0.0182	0.0478	0.0553	0.0575	*0.0637	0.2071	0.2517	0.2546
0.7	*0.0609	0.1011	0.1161	0.1206	*0.1282	0.2511	0.3050	0.3092
0.9	*0.2022	0.2401	0.2544	0.2594	*0.2628	0.3434	0.3819	0.3902
	$(\Delta_1, l_2, \Delta_2, l_2) = (3.5, 50, 1, 1000)$ and $p - l_1 - l_2$ $\Delta' s \sim N(0, 0.1^2)$							
	$p = 10^4$				$p = 10^5$			
ρ	EB	CMLE	FAIR	IR	EB	CMLE	FAIR	IR
0.3	*0.0030	0.0162	0.0297	0.0305	*0.0173	0.0647	0.2169	0.2183
0.5	*0.0125	0.0357	0.0564	0.0590	*0.0408	0.0879	0.2520	0.2552
0.7	*0.0501	0.0856	0.1165	0.1215	*0.0966	0.1410	0.3051	0.3095
0.9	*0.1825	0.2143	0.2554	0.2612	*0.2397	0.2748	0.3880	0.3913

Table 2: Dependent case I : $Corr(X_i, X_j) = \rho_{ij} = \rho^{|i-j|}$ for $\rho = 0.3, 0.5$ and 0.7. $(\Delta_1, l_1, \Delta_2, l_2)$ represents l_1 and l_2 coordinates in ν are Δ_1 and Δ_2 respectively.

produce smaller error rates compared to FAIR and IR. However, compared to the results in Table 1, the EB method and CMLE have a worst performance which is caused by some sensitivity to the heavy tailed distribution of the X_js.

Summary: The most important advantage of the EB classifier, demonstrated in the above simulations, is its ability to use the information provided by many small signals in order to improve the classification. This is unlike variable-selection type of classifiers, that give up on using the information from variables with small ν_j, in order to reduce the variability in estimation. This advantage is not on the expanse of being a good classifier also under moderately sparse configurations.

4.2 Real Data Analysis

The following analysis of real date sets is based on the procedure described in Section 3.2. We consider three real data sets and compare the empirical Bayes approach with nearest centroid shrunken (henceforth NSC), and FAIR. The NSC was proposed by Tibshirani et al. (2002). The three data sets were studied by Fan and Fan (2008), and all the misclassification rates, other than that of the empirical Bayes method, are cited from that paper.

The first example is of a leukemia data set, which was previously analyzed in Golub et al. (1999). The data set can be obtained in `http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi`. There are $p = 7129$ genes and 72 samples generated from two classes, ALL (acute lymphocytic leukemia) and AML (acute mylogenous leukemia). Among the 72 samples, the training data

	$(\Delta, l) = (1, 2000)$, $p - l$ $\Delta \sim N(0, 0.1^2)$							
	$p = 10^4$				$p = 10^5$			
a	EB	CMLE	FAIR	IR	EB	CMLE	FAIR	IR
0.3	*0.0326	0.1114	0.1138	0.1150	*0.2756	0.4160	0.4058	0.4075
0.5	*0.1415	0.2405	0.2468	0.2487	*0.3059	0.4426	0.4438	0.4453
0.7	*0.1947	0.3145	0.3243	0.3284	*0.4038	0.4582	0.4783	0.4795
0.9	*0.2370	0.3586	0.3889	0.3961	0.4868	*0.4583	0.4817	0.4837
	$(\Delta_1, l_1, \Delta_2, l_2) = (2.5, 100, 1, 1000)$ and $p - l_1 - l_2, \Delta' s \sim N(0, 0.1^2)$							
	$p = 10^4$				$p = 10^5$			
a	EB	CMLE	FAIR	IR	EB	CMLE	FAIR	IR
0.3	*0.0585	0.1051	0.1577	0.1639	*0.2712	0.2757	0.4102	0.4127
0.5	*0.1940	0.2225	0.3099	0.3156	0.3532	*0.3123	0.4669	0.4682
0.7	*0.2468	0.2588	0.3684	0.3760	0.4181	*0.3402	0.4781	0.4795
0.9	*0.2803	0.2543	0.3997	0.4093	0.4765	*0.3418	0.4815	0.4828
	$(\Delta_1, l_1, \Delta_2, l_2) = (3.5, 50, 1, 1000)$ and $p - l_1 - l_2$ $\Delta' s \sim N(0, 0.1^2)$							
	$p = 10^4$				$p = 10^5$			
a	EB	CMLE	FAIR	IR	EB	CMLE	FAIR	IR
0.3	0.0479	*0.0469	0.1672	0.1740	0.2656	*0.1481	0.4189	0.4215
0.5	0.1711	*0.1272	0.3064	0.3125	0.2883	*0.1617	0.4513	0.4646
0.7	0.1730	*0.1300	0.3540	0.3629	0.3890	*0.1801	0.4820	0.4835
0.9	0.2486	*0.1424	0.3756	0.3882	0.4794	*0.2167	0.4867	0.4882

Table 3: Dependent case II : $Corr(X_i, X_j) = \rho_{ij} = \alpha_i \alpha_j$ for $i \neq j$ where α_i and α_j are generated from $Unif(-a, a)$ for $a = 0.3, 0.5, 0.7$ and 0.9. $(\Delta_1, l_1, \Delta_2, l_2)$ represents l_1 and l_2 coordinates in ν are Δ_1 and Δ_2 respectively.

set has 38 ($n_1 = 27$ in ALL and $n_2 = 11$ in AML) and the test data set has 34 (20 in ALL and 14 in AML). Table 6 shows the results of the nearest shrunken centroid, FAIR, and empirical Bayes methods.

The empirical Bayes approach misclassified 3 out of 34 test samples which is the same result as NSC, but slightly worse than FAIR. Figure 1 shows histograms of $\sum_j \hat{a}_j U_j$ corresponding to the two groups, under the training and under the test sets.

The second example is of lung cancer data which were previously analyzed by Gordon et al. (2002) and analyzed using FAIR in Fan and Fan (2008). The data is available at `http://www.chestsurg.org`. There are $p = 12533$ genes and 181 samples coming from two classes, MPM(malignant pleural mesothelioma) and ADCA(adenocarcinoma). The training sample set consists of 32 samples($n_1 = 16$ from MPM and $n_2 = 16$ from ADCA) and the test has 149 samples (15 from MPM and 134 from ADCA). As displayed in Table 7, the empirical Bayes method classified all the training samples correctly and 148 out of 149 test samples correctly, which is a significant improvement compared to NSC and FAIR. In Figure 2, we show histograms of $\sum \hat{a}_j U_j$ under the two groups, for the training and for the test sets.

The last example is of prostate cancer data studied by Singh et al. (2002), which is available at `http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi`. The training data set has 102 samples, $n_1 = 52$ of which are prostate tumor samples and $n_2 = 50$ of which are normal samples. An

	$p = 10^4$, $p-l$ Δ's are 0			
(Δ, l)	EB	CMLE	FAIR	IR
$(0.2,\ 10^4)$	*0.0066	0.4103	0.2900	0.2896
$(0.1,\ 10^4)$	*0.1678	0.4831	0.4448	0.4447
$(0.2,\ 5\times 10^3)$	*0.1546	0.4594	0.3905	0.3902
$(0.1,\ 5\times 10^3)$	*0.3725	0.4931	0.4720	0.4719
	$p = 10^5$, $p-l$ Δ's are 0			
(Δ, l)	EB	CMLE	FAIR	IR
$(0.2,\ 10^5)$	*0.0000	0.0947	0.0415	0.0411
$(0.1,\ 10^5)$	*0.0004	0.4437	0.3301	0.3297
$(0.2,\ 5\times 10^4)$	*0.0002	0.3314	0.1907	0.1902
$(0.1,\ 5\times 10^4)$	*0.1181	0.4779	0.4128	0.4126

Table 4: Non-sparse case

	$p = 10^4$							
	$p-l$ Δ's are 0				$p-l$ Δ's $\sim N(0, 0.1^2)$			
(Δ, l)	EB	CMLE	FAIR	IR	EB	CMLE	FAIR	IR
$(1.0,\ 2000)$	0.0350	0.1855	0.0152	*0.0149	0.0267	0.2126	0.0084	*0.0081
$(2.5,\ 100)$	0.1888	*0.1639	0.2121	0.2123	*0.1576	0.1600	0.1959	0.1969
$(3.5,\ 50)$	0.0994	*0.0791	0.2111	0.2112	0.2045	*0.1967	0.2643	0.2650
$(4.0\ ,40)$	0.0744	*0.0640	0.2050	0.2055	0.0714	*0.0681	0.1933	0.1939
	$p = 10^5$							
	$p-l$ Δ's are 0				$p-l$ Δ's $\sim N(0, 0.1^2)$			
(Δ, l)	EB	CMLE	FAIR	IR	EB	CMLE	FAIR	IR
$(1.0,\ 2000)$	0.4331	0.4385	0.2179	*0.2170	0.2667	0.4325	*0.1886	0.1897
$(2.5,\ 100)$	0.4335	0.4155	*0.4018	0.4018	*0.3115	0.3926	0.3518	0.3517
$(3.5,\ 50)$	0.3661	*0.3365	0.4027	0.4023	*0.2966	0.3560	0.3562	0.3562
$(4.0\ ,40)$	0.3528	*0.3282	0.4017	0.4023	*0.2815	0.3202	0.3544	0.3542

Table 5: Heavy tail case.

Method	Training error	Test error
Nearest shrunken centroids	1/38	3/34
FAIR	1/38	1/34
E.B.	0/38	3/34

Table 6: Classification errors of Leukemia data set

Method	Training error	Test error
Nearest shrunken centroids	0/32	11/149
FAIR	0/32	7/149
E.B.	0/32	1/149

Table 7: Classification errors of Lung Cancer data set

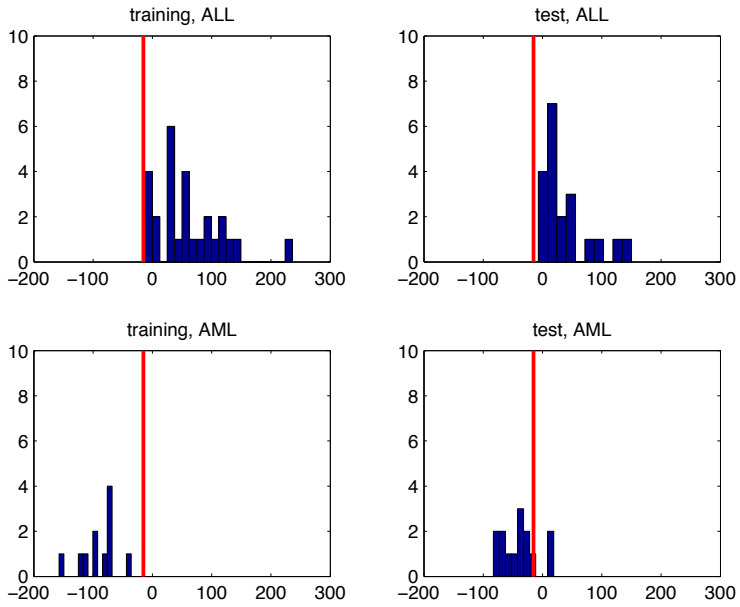


Figure 1: Histograms of $\sum_j \hat{a}_j U_j$ of ALL and AML for training and test sets of Leukemia data. Two panels in the first columns are histograms for ALL and AML from training sets and two in the second columns are for ALL and AML from test sets. Red vertical lines in all histograms represent cut off value which is $-\hat{a}_0 = (\hat{\theta}_{ALL} + \hat{\theta}_{AML})/2 = -15.10$

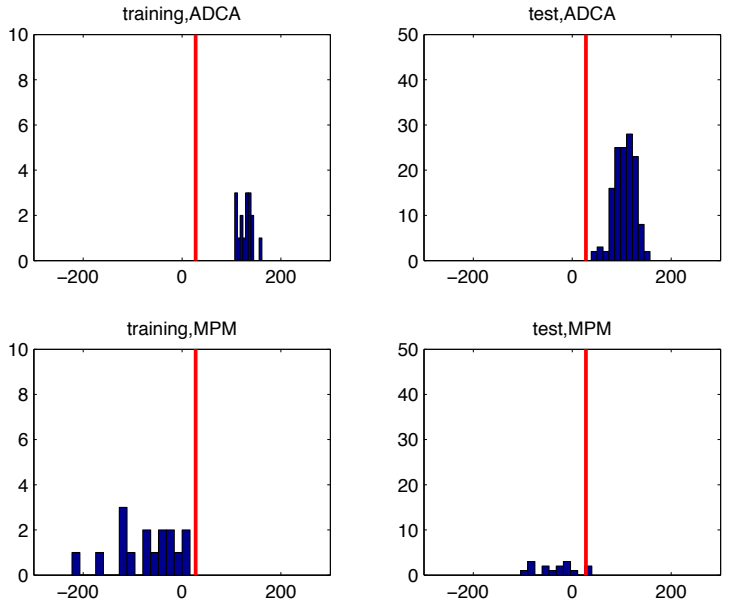


Figure 2: Histograms of $\sum_j \hat{a}_j U_j$ of ADCA and MPM for training and test sets of lung cancer data. Two panels in the first columns are histograms for ADCA and MPM from training sets and two in the second columns are for ADCA and MPM from test sets. Red vertical lines in all histograms represent cut off value which is $-a_0 = (\hat{\theta}_{ADCA} + \hat{\theta}_{MPM})/2 = 27.54$.

Method	Training error	Test error
Nearest shrunken centroids	8/102	9/34
FAIR	10/102	9/34
E.B.	38/102	4/34

Table 8: Classification errors of Prostate Cancer data set

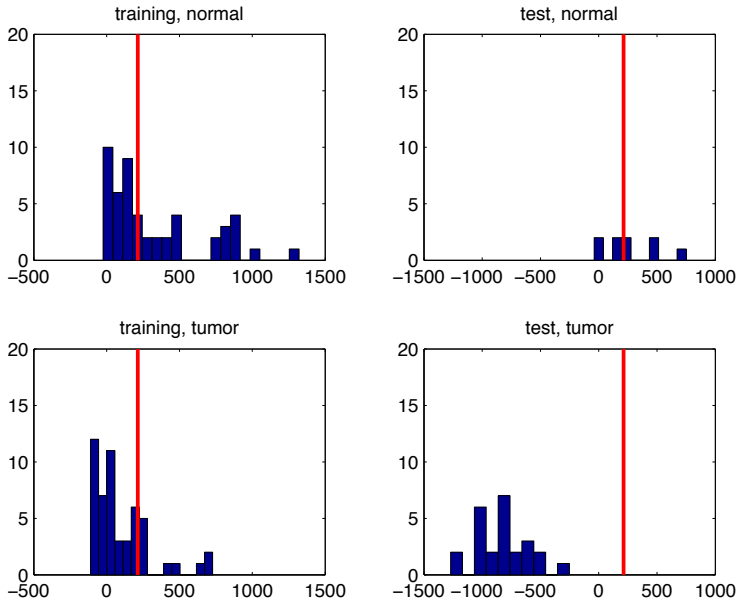


Figure 3: Histograms of $\sum_j \hat{a}_j U_j$ of normal and tumor for training and test sets of prostate cancer data. Two panels in the first columns are histograms for normal and tumor from training sets and two in the second columns are for normal and tumor from test sets. Red vertical lines in all histograms represent cut off value which is $-a_0 = (\hat{\theta}_{normal} + \hat{\theta}_{tumor})/2 = 213.68$.

independent test data set, from a different experiment, has 25 tumor and 9 normal samples. There are $p = 12600$ genes.

As displayed in Table 8, for the prostate cancer data, the empirical Bayes approach has a very large training error compared to NSC and FAIR, but the test error is smaller than both NSC and FAIR. The *pessimism* of the misclassification error, reflected by our training set, may be attributed to two facts. One is the difference in the proportion of tumor and normal samples in the training versus the test set. The other reason is that the test set seems to be less noisy. It seems that the empirical Bayes method succeed in estimating ν_j and hence deriving good coefficients $\hat{a}_j$ from the large training data although it is noisy; yet, the classification of the individual data points of the noisy training set is still difficult, while the classification is easier for the test set data points. Figure 3 might be helpful in assessing it. In the histograms of $\sum_j \hat{a}_j U_j$ corresponding to the normal and tumor groups from the training data, we may see that the two training sets look noisier.

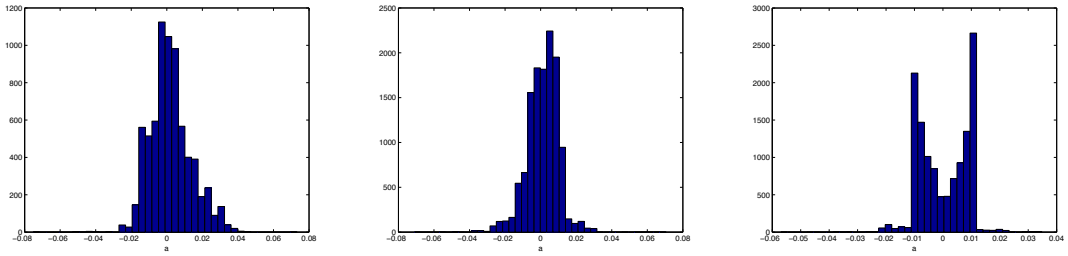


Figure 4: Histograms of $\hat{a}_j$, $j = 1, ..., p$, for the leukemia, lung cancer, and prostate cancer data sets.

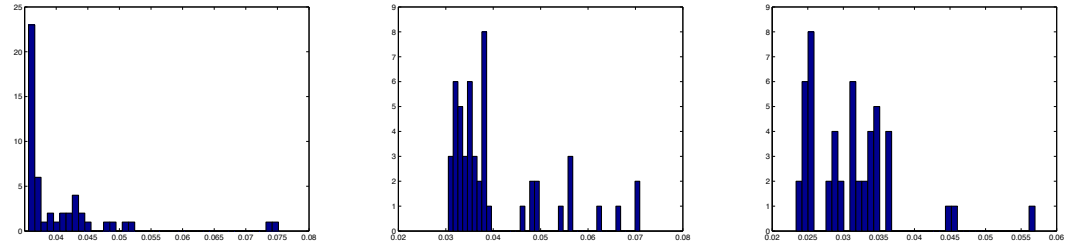


Figure 5: Histograms of the first 50 largest $|\hat{a}_j|$ for the leukemia, lung cancer, and prostate cancer data sets.

4.3 Number of Selected Variables

Figure 4 shows the histograms of $\hat{a}_j$, $j = 1, ..., p$ for each data set. In Figure 5 we see three histograms corresponding to the fifty largest $|\hat{a}_j|$, $j = 1, ..., p$, in each of the three data sets. Our empirical Bayes method uses many variables for the classification. In fact, formally it uses all the variables, since none of the $\hat{a}_j$ is exactly 0. In comparison the FAIR uses 11, 31, and 2 variables corresponding to the above three cases in the order they presented, while the NSC uses 21, 26, 6.

Obviously a method which is based on a few variables is easy to implement and to interpret. Our suggested classifiers are meant only to produce good classification and thus use many variables if necessary. Using many variables and somewhat complicated classifiers is in the spirit of data mining approach. However, selecting a subset of variables following an empirical Bayes estimation of the means, makes much sense, for producing simpler classifiers. It might even reduce noise and will produce over all better classifiers.

Acknowledgments

We are grateful to Yingying Fan for providing us some of the data analyzed in Fan and Fan (2008). We also thanks the AE and three reviewers for helpful comments for improvement of the paper.

References

P.J. Bickel and E. Levina. Some theory for Fisher's linear discriminant function, "naive Bayes", and some alternatives where there are many more variables than observations. *Bernoulli*, 10(6):989-1010, 2004.

L.D. Brown. Admissible estimators, recurrent diffusions, and insoluble boundary value problems. *The Annals of Mathematical Statistics*, 42(3):855-903, 1971.

L.D. Brown. In-season prediction of batting averages: a field-test of simple empirical Bayes and Bayes methodologies. *Annals of Applied Statistics*, 2(1):113-152, 2008.

L.D. Brown and E. Greenshtein. Nonparametric empirical Bayes and compound decision approaches to estimation of a high-dimensional vector of means. *Annals of Statistics*, 37(4):1685-1704, 2009.

J.B. Copas. Compound decisions and empirical Bayes. *Journal of the Royal Statistical Society Series B(Methodological)*, 31(3):397-425, 1969.

B. Efron. Empirical Bayes estimates for large-scale prediction problems. *Journal of the American Statistical Association*, forthcoming.

J. Fan and Y. Fan. High dimensional classification using features annealed independence rules. *Annals of Statistics*, 36(6):2605-2637, 2008.

E. Greenshtein, J. Park, and G. Lebannon. Regularization through variable selection and conditional mle with application to classification in high dimensions. *Journal of Statistical Planning and Inference*, 139(2):385-395, 2009.

E. Greenshtein and Y. Ritov. Asymptotic efficiency of simple decisions for the compound decision problem. *The Third Lehmann Symposium, IMS Lecture Notes Monograph Series*, forthcoming.

T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531-537, 1999.

G.J. Gordon, R.V. Jensen, L.L. Hsiao, S.R. Gullans, J.E. Blumenstock, S. Ramaswamy, W.G. Richards, D.J. Sugarbaker, and R. Bueno. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Res*, 62(17):4963-4967, 2002.

W. Jiang and C.H. Zhang. General maximum likelihood empirical Bayes estimation of normal means. *Annals of Statistics*, 37(4):1647-1684, 2009.

E.L. Lehmann. *Testing Statistical Hypothesis*. Wiley, 1986.

H. Robbins. Asymptotically subminimax solutions of compound decision problems. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 131-148, Berkeley, California, 1951.

D. Singh, P.G. Febbo, K. Ross, D.G. Jackson, J. Manola, C. Ladd, P. Tamayo, A.A. Renshaw, A.V. D'Amico, J.P. Richie, E.S. Lander, M. Loda, P.W. Kantoff, T.R. Golub, W.R. Sellers. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1(2):203-209, 2002.

R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. In *Proceedings of National Academy of Sciences*, 99(1):6567-6572, 2002.

C.H. Zhang. Compound decision theory and empirical Bayes methods. *Annals of Statistics*, 31(2):379-390, 2003.

Journal of Machine Learning Research 2009 (10) 1705-1736 Submitted 9/08; Published 7/09

Learning Permutations with Exponential Weights*

David P. Helmbold DPH@CSE.UCSC.EDU
Manfred K. Warmuth† MANFRED@CSE.UCSC.EDU
Computer Science Department
University of California, Santa Cruz
Santa Cruz, CA 95064

Editor: Yoav Freund

Abstract

We give an algorithm for the on-line learning of permutations. The algorithm maintains its uncertainty about the target permutation as a doubly stochastic weight matrix, and makes predictions using an efficient method for decomposing the weight matrix into a convex combination of permutations. The weight matrix is updated by multiplying the current matrix entries by exponential factors, and an iterative procedure is needed to restore double stochasticity. Even though the result of this procedure does not have a closed form, a new analysis approach allows us to prove an optimal (up to small constant factors) bound on the regret of our algorithm. This regret bound is significantly better than that of either Kalai and Vempala's more efficient Follow the Perturbed Leader algorithm or the computationally expensive method of explicitly representing each permutation as an expert.

Keywords: permutation, ranking, on-line learning, Hedge algorithm, doubly stochastic matrix, relative entropy projection, Sinkhorn balancing

1. Introduction

Finding a good permutation is a key aspect of many problems such as the ranking of search results or matching workers to tasks. In this paper we present an efficient and effective on-line algorithm for learning permutations in a model related to the on-line allocation model of learning with experts (Freund and Schapire, 1997). In each trial, the algorithm probabilistically chooses a permutation and then incurs a linear loss based on how appropriate the permutation was for that trial. The *regret* is the total expected loss of the algorithm on the whole sequence of trials minus the total loss of the best permutation chosen in hindsight for the whole sequence, and the goal is to find algorithms that have provably small worst-case regret.

For example, one could consider a commuter airline which owns n airplanes of various sizes and flies n routes.[1] Each day the airline must match airplanes to routes. If too small an airplane is assigned to a route then the airline will loose revenue and reputation due to unserved potential passengers. On the other hand, if too large an airplane is used on a long route then the airline could have larger than necessary fuel costs. If the number of passengers wanting each flight were known ahead of time, then choosing an assignment is a weighted matching problem. In the on-line

*. An earlier version of this paper appears in *Proceedings of the Twentieth Annual Conference on Computational Learning Theory* (COLT 2007), published by Springer as LNAI 4539.
†. Manfred K. Warmuth acknowledges the support of NSF grant IIS 0325363.
1. We assume that each route starts and ends at the airline's home airport.

allocation model, the airline first chooses a distribution over possible assignments of airplanes to routes and then randomly selects an assignment from the distribution. The *regret* of the airline is the earnings of the single best assignment for the whole sequence of passenger requests minus the total expected earnings of the on-line assignments. When airplanes and routes are each numbered from 1 to n, then an assignment is equivalent to selecting a permutation. The randomness helps protect the on-line algorithm from adversaries and allows one to prove good bounds on the algorithm's regret for arbitrary sequences of requests.

Since there are $n!$ permutations on n elements, it is infeasible to simply treat each permutation as an expert and apply one of the expert algorithms that uses exponential weights. Previous work has exploited the combinatorial structure of other large sets of experts to create efficient algorithms (see Helmbold and Schapire, 1997; Takimoto and Warmuth, 2003; Warmuth and Kuzmin, 2008, for examples). Our solution is to make a simplifying assumption on the loss function which allows the new algorithm, called PermELearn, to maintain a sufficient amount of information about the distribution over $n!$ permutations while using only n^2 weights.

We represent a permutation of n elements as an $n \times n$ *permutation matrix* Π where $\Pi_{i,j} = 1$ if the permutation maps element i to position j and $\Pi_{i,j} = 0$ otherwise. As the algorithm randomly selects a permutation $\widehat{\Pi}$ at the beginning of a trial, an adversary simultaneously selects an arbitrary *loss matrix* $L \in [0,1]^{n \times n}$ which specifies the loss of all permutations for the trial. Each entry $L_{i,j}$ of the loss matrix gives the loss for mapping element i to j, and the loss of any whole permutation is the sum of the losses of the permutation's mappings, that is, the loss of permutation Π is $\sum_i L_{i,\Pi(i)} = \sum_{i,j} \Pi_{i,j} L_{i,j}$. Note that the per-trial expected losses can be as large as n, as opposed to the common assumption for the expert setting that the losses are bounded in $[0,1]$. In Section 3 we show how a variety of intuitive loss motifs can be expressed in this matrix form.

This assumption that the loss has a linear matrix form ensures the expected loss of the algorithm can be expressed as $\sum_{i,j} W_{i,j} L_{i,j}$, where $W = \mathbb{E}(\widehat{\Pi})$. This expectation W is an $n \times n$ *weight matrix* which is *doubly stochastic*, that is, it has non-negative entries and the property that every row and column sums to 1. The algorithm's uncertainty about which permutation is the target is summarized by W; each weight $W_{i,j}$ is the probability that the algorithm predicts with a permutation mapping element i to position j. It is worth emphasizing that the W matrix is only a *summary* of the distribution over permutations used by any algorithm (it doesn't indicate which permutations have non-zero probability, for example). However, this summary is *sufficient* to determine the algorithm's expected loss when the losses of permutations have the assumed loss matrix form.

Our PermELearn algorithm stores the weight matrix W and must convert W into an efficiently sampled distribution over permutations in order to make predictions. By Birkhoff's Theorem, every doubly stochastic matrix can be expressed as the convex combination of at most $n^2 - 2n + 2$ permutations (see, e.g., Bhatia, 1997). In Appendix A we show that a greedy matching-based algorithm efficiently decomposes any doubly stochastic matrix into a convex combination of at most $n^2 - 2n + 2$ permutations. Although the efficacy of this algorithm is implied by standard dimensionality arguments, we give a new combinatorial proof that provides independent insight as to why the algorithm finds a convex combination matching Birkhoff's bound. Our algorithm for learning permutations predicts with a random $\widehat{\Pi}$ sampled from the convex combination of permutations created by decomposing weight matrix W. It has been applied recently for pricing combinatorial markets when the outcomes are permutations of objects (Chen et al., 2008).

The PermELearn algorithm updates the entries of its weight matrix using exponential factors commonly used for updating the weights of experts in on-line learning algorithms (Littlestone and

Warmuth, 1994; Vovk, 1990; Freund and Schapire, 1997): each entry $W_{i,j}$ is multiplied by a factor $e^{-\eta L_{i,j}}$. Here η is a positive learning rate that controls the "strength" of the update (When $\eta = 0$, than all the factors are one and the update is vacuous). After this update, the weight matrix no longer has the doubly stochastic property, and the weight matrix must be projected back into the space of doubly stochastic matrices (called "Sinkhorn balancing", see Section 4) before the next prediction can be made.

In Theorem 4 we bound the expected loss of PermELearn over any sequence of trials by

$$\frac{n \ln n + \eta \mathcal{L}_{\text{best}}}{1 - e^{-\eta}}, \tag{1}$$

where n is the number of elements being permuted, η is the learning rate, and $\mathcal{L}_{\text{best}}$ is the loss of the best permutation on the entire sequence. If an upper bound $\mathcal{L}_{\text{est}} \geq \mathcal{L}_{\text{best}}$ is known, then η can be tuned (as in Freund and Schapire, 1997) and the expected loss bound becomes

$$\mathcal{L}_{\text{best}} + \sqrt{2\mathcal{L}_{\text{est}} n \ln n} + n \ln n, \tag{2}$$

giving a bound of $\sqrt{2\mathcal{L}_{\text{est}} n \ln n} + n \ln n$ on the worst case expected regret of the tuned PermELearn algorithm. We also prove a matching lower bound (Theorem 6) of $\Omega(\sqrt{\mathcal{L}_{\text{best}} n \ln n})$ for the expected regret of any algorithm solving our permutation learning problem.

A simpler and more efficient algorithm than PermELearn maintains the sum of the loss matrices on the the previous trials. Each trial it adds random perturbations to the cumulative loss matrix and then predicts with the permutation having minimum perturbed loss. This "Follow the Perturbed Leader" algorithm (Kalai and Vempala, 2005) has good regret bounds for many on-line learning settings. However, the regret bound we can obtain for it in the permutation setting is about a factor of n worse than the bound for PermELearn and the lower bound.

Although computationally expensive, one can also consider running the Hedge algorithm while explicitly representing each of the $n!$ permutations as an expert. If T is the sum of the loss matrices over the past trials and F is the $n \times n$ matrix with entries $F_{i,j} = e^{-\eta T_{i,j}}$, then the weight of each permutation expert Π is proportional to the product $\prod_i F_{i,\Pi(i)}$ and the normalization constant is the permanent of the matrix F. Calculating the permanent is a known #P-complete problem and sampling from this distribution over permutations is very inefficient (Jerrum et al., 2004). Moreover since the loss range of a permutation is $[0, n]$, the standard loss bound for the algorithm that uses one expert per permutation must be scaled up by a factor of n, becoming

$$\mathcal{L}_{\text{best}} + n\sqrt{2\frac{\mathcal{L}_{\text{est}}}{n} \ln(n!)} + n \ln(n!) \approx \mathcal{L}_{\text{best}} + \sqrt{2\mathcal{L}_{\text{est}} n^2 \ln n} + n^2 \ln n.$$

This expected loss bound is similar to our expected loss bound for PermELearn in Equation (2), except that the $n \ln n$ terms are replaced by $n^2 \ln n$. Our method based on Sinkhorn balancing bypasses the estimation of permanents and somehow PermELearn's implicit representation and prediction method exploit the structure of permutations and lets us obtain the improved bound. We also give a matching lower bound that shows PermELearn has the optimum regret bound (up to a small constant factor). It is an interesting open question whether the structure of permutations can be exploited to prove bounds like (2) for the Hedge algorithm with one expert per permutation.

PermELearn's weight updates belong to the Exponentiated Gradient family of updates (Kivinen and Warmuth, 1997) since the components $L_{i,j}$ of the loss matrix that appear in the exponential

factor are the derivatives of our linear loss with respect to the weights $W_{i,j}$. This family of updates usually maintains a probability vector as its weight vector. In that case the normalization of the weight vector is straightforward and is folded directly into the update formula. Our new algorithm PermELearn for learning permutations maintains a doubly stochastic matrix with n^2 weights. The normalization alternately normalizes the rows and columns of the matrix until convergence (Sinkhorn balancing). This may require an unbounded number of steps and the resulting matrix does not have a closed form. Despite this fact, we are able to prove bounds for our algorithm.

We first show that our update minimizes a tradeoff between the loss and a relative entropy between doubly stochastic matrices. This relative entropy becomes our measure of progress in the analysis. Luckily, the un-normalized multiplicative update already makes enough progress (towards the best permutation) to achieve the loss bound quoted above. Finally, we interpret the iterations of Sinkhorn balancing as Bregman projections with respect to the same relative entropy and show using the properties of Bregman projections that these projections can only increase the progress and thus don't hurt the analysis (Herbster and Warmuth, 2001).

Our new insight of splitting the update into an un-normalized step followed by a normalization step also leads to a streamlined proof of the loss bound for the Hedge algorithm in the standard expert setting that is interesting in its own right. Since the loss in the allocation setting is linear, the bounds can be proven in many different ways, including potential based methods (see, e.g., Kivinen and Warmuth, 1999; Gordon, 2006; Cesa-Bianchi and Lugosi, 2006). For the sake of completeness we reprove our main loss bound for PermELearn using potential based methods in Appendix B. We show how potential based proof methods can be extended to handle linear equality constraints that don't have a solution in closed form, paralleling a related extension to linear *in*equality constraints in Kuzmin and Warmuth (2007). In this appendix we also discuss the relationship between the projection and potential based proof methods. In particular, we show how the Bregman projection step corresponds to plugging in suboptimal dual variables into the potential.

The remainder of the paper is organized as follows. We introduce our notation in the next section. Section 3 presents the permutation learning model and gives several intuitive examples of appropriate loss motifs. Section 4 gives the PermELearn algorithm and discusses its computational requirements. One part of the algorithm is to decompose the current doubly stochastic matrix into a small convex combination of permutations using a greedy algorithm. The bound on the number of permutations needed to decompose the weight matrix is deferred to Appendix A. We then bound PermELearn's regret in Section 5 in a two-step analysis that uses a relative entropy as a measure of progress. To exemplify the new techniques, we also analyze the basic Hedge algorithm with the same methodology. The regret bounds for Hedge and PermELearn are re-proven in Appendix B using potential based methods. In Section 6, we apply the "Follow the Perturbed Leader" algorithm to learning permutations and show that the resulting regret bounds are not as good. In Section 7 we prove a lower bound on the regret when learning permutations that is within a small constant factor of our regret bound on the tuned PermELearn algorithm. The concluding section describes extensions and directions for further work.

2. Notation

All matrices will be $n \times n$ matrices. When A is a matrix, $A_{i,j}$ denotes the entry of A in row i, and column j. We use $A \bullet B$ to denote the dot product between matrices A and B, that is, $\sum_{i,j} A_{i,j} B_{i,j}$. We use single superscripts (e.g., A^k) to identify matrices/permutations from a sequence.

Permutations on n elements are frequently represented in two ways: as a bijective mapping of the elements $\{1,\ldots,n\}$ into the positions $\{1,\ldots,n\}$ or as a permutation matrix which is an $n \times n$ binary matrix with exactly one "1" in each row and each column. We use the notation Π (and $\widehat{\Pi}$) to represent a permutation in either format, using the context to indicate the appropriate representation. Thus, for each $i \in \{1,\ldots,n\}$, we use $\Pi(i)$ to denote the position that the ith element is mapped to by permutation Π, and matrix element $\Pi_{i,j} = 1$ if $\Pi(i) = j$ and 0 otherwise.

If L is a matrix with n rows then the product ΠL permutes the rows of L:

$$\underset{\text{perm. } (2,4,3,1) \text{ as matrix}}{\Pi = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}} \qquad \underset{\text{an arbitrary matrix}}{L = \begin{pmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{pmatrix}} \qquad \underset{\text{permuting the rows}}{\Pi L = \begin{pmatrix} 21 & 22 & 23 & 24 \\ 41 & 42 & 43 & 44 \\ 31 & 32 & 33 & 34 \\ 11 & 12 & 13 & 14 \end{pmatrix}}.$$

Convex combinations of permutations create *doubly stochastic* or *balanced* matrices: non-negative matrices whose n rows and n columns each sum to one. Our algorithm maintains its uncertainty about which permutation is best as a doubly stochastic weight matrix W and needs to randomly select a permutation from some distribution whose expectation is W. By Birkhoff's Theorem (see, e.g., Bhatia, 1997), for every doubly stochastic matrix W there is a decomposition into a convex combination of at most $n^2 - 2n + 2$ permutation matrices. We show in Appendix A how a decomposition of this size can be found effectively. This decomposition gives a distribution over permutations whose expectation is W that now can be effectively sampled because its support is at most $n^2 - 2n + 2$ permutations.

3. On-line Protocol

We are interested in learning permutations in a model related to the on-line allocation model of learning with experts (Freund and Schapire, 1997). In that model there are N experts and at the beginning of each trial the algorithm allocates a probability distribution w over the experts. The algorithm picks expert i with probability w_i and then receives a loss vector $\ell \in [0,1]^N$. Each expert i incurs loss ℓ_i and the expected loss of the algorithm is $w \cdot \ell$. Finally, the algorithm updates its distribution w for the next trial.

In case of permutations we could have one expert per permutation and allocate a distribution over the $n!$ permutations. Explicitly tracking this distribution is computationally expensive, even for moderate n. As discussed in the introduction, we assume that the losses in each trial can be specified by a loss matrix $L \in [0,1]^{n \times n}$ where the loss of each permutation Π has the linear form $\sum_i L_{i,\Pi(i)} = \Pi \bullet L$. If the algorithm's prediction $\widehat{\Pi}$ is chosen probabilistically in each trial then the algorithm's expected loss is $\mathbb{E}[\widehat{\Pi} \bullet L] = W \bullet L$, where $W = \mathbb{E}[\widehat{\Pi}]$. This expected prediction W is an $n \times n$ doubly stochastic matrix and algorithms for learning permutations under the linear loss assumption can be viewed as implicitly maintaining such a doubly stochastic weight matrix.

More precisely, the on-line algorithm follows the following protocol in each trial:

- The learner (probabilistically) chooses a permutation $\widehat{\Pi}$, and let $W = \mathbb{E}(\widehat{\Pi})$.
- Nature simultaneously chooses a loss matrix $L \in [0,1]^{n \times n}$ for the trial.
- At the end of the trial, the algorithm is given L. The loss of $\widehat{\Pi}$ is $\widehat{\Pi} \bullet L$ and the expected loss of the algorithm is $W \bullet L$.

- Finally, the algorithm updates its distribution over permutations for the next trial, implicitly updating matrix W.

Although our algorithm can handle arbitrary sequences of loss matrices $L \in [0,1]^{n\times n}$, nature could be significantly more restricted. Many ranking applications have an associated loss motif M and nature is constrained to choose (row) permutations of M as its loss matrix L. In effect, at each trial nature chooses a "correct" permutation Π and uses the loss matrix $L = \Pi M$. Note that the permutation left-multiplies the loss motif, and thus permutes the rows of M. If nature chooses the identity permutation then the loss matrix L is the motif M itself. When M is known to the algorithm, it suffices to give the algorithm only the permutation Π at the end of the trial, rather than the loss matrix L itself. Figure 1 gives examples of loss motifs.

The last loss in Figure 1 is related to a competitive List Update Problem where an algorithm services requests to a list of n items. In the List Update Problem the cost of a request is the requested item's current position in the list. After each request, the requested item can be moved forward in the list for free, and additional rearrangement can be done at a cost of one per transposition. The goal is for the algorithm to be cost-competitive with the best static ordering of the elements in hindsight. Note that the transposition cost for additional list rearrangement is not represented in the permutation loss motif. Blum et al. (2003) give very efficient algorithms for the List Update Problem that do not do additional rearranging of the list (and thus do not incur the cost neglect by the loss motif). In our notation, their bound has the same form as ours (1) but with the $n\ln n$ factors replaced by $O(n)$. However, our lower bound (see Section 7) shows that the $n\ln n$ factors in (2) are necessary in the general permutation setting.

Note that many compositions of loss motifs are possible. For example, given two motifs with their associated losses, any convex combination of the motifs creates a new motif for the same convex combination of the associated losses. Other component-wise combinations of two motifs (such as product or max) can also produce interesting loss motifs, but the combination usually cannot be distributed across the matrix dot-product calculation, and so cannot be expressed as a simple linear function of the original losses.

4. PermELearn Algorithm

Our permutation learning algorithm uses exponenential weights and we call it PermELearn. It maintains an $n \times n$ doubly stochastic weight matrix W as its main data structure, where $W_{i,j}$ is the probability that PermELearn predicts with a permutation mapping element i to position j. In the absence of prior information it is natural to start with uniform weights, that is, the matrix with $\frac{1}{n}$ in each entry.

In each trial PermELearn does two things:

1. Choose a permutation $\widehat{\Pi}$ from some distribution such that $\mathbb{E}[\widehat{\Pi}] = W$.

2. Create a new doubly stochastic matrix $\widetilde{W}$ for use in the next trial based on the current weight matrix W and loss matrix L.

loss $\mathcal{L}(\widehat{\Pi}, \Pi)$	motif M
the number of elements i where $\widehat{\Pi}(i) \neq \Pi$	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$
$\frac{1}{n-1}\sum_{i=1}^{n} \lvert\widehat{\Pi}(i) - \Pi(i)\rvert$, how far the elements are from their "correct" positions (the division by $n-1$ ensures that the entries of M are in $[0,1]$.)	$\frac{1}{3}\begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{pmatrix}$
$\frac{1}{n-1}\sum_{i=1}^{n} \frac{\lvert\widehat{\Pi}(i)-\Pi(i)\rvert}{\Pi(i)}$, a position weighted version of the above emphasizing the early positions in Π	$\frac{1}{3}\begin{pmatrix} 0 & 1 & 2 & 3 \\ 1/2 & 0 & 1/2 & 1 \\ 2/3 & 1/3 & 0 & 1/3 \\ 3/4 & 1/2 & 1/4 & 0 \end{pmatrix}$
the number of elements mapped to the first half by Π but the second half by $\widehat{\Pi}$, or vice versa	$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$
the number of elements mapped to the first two positions by Π that fail to appear in the top three position of $\widehat{\Pi}$	$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
the number of links traversed to find the first element of Π in a list ordered by $\widehat{\Pi}$	$\frac{1}{3}\begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

Figure 1: Loss motifs

Choosing a permutation is done by Algorithm 1. The algorithm greedily decomposes W into a convex combination of at most $n^2 - 2n + 2$ permutations (see Theorem 7), and then randomly selects one of these permutations for the prediction.[2]

Our decomposition algorithm uses a Temporary matrix A initialized to the weight matrix W. Each iteration of Algorithm 1 finds a permutation Π where each $A_{i,\Pi(i)} > 0$. This can be done by finding a perfect matching on the $n \times n$ bipartite graph containing the edge i, j whenever $A_{i,j} > 0$. We shall soon see that each matrix A is a constant times a doubly stochastic matrix, so the existence of a suitable permutation Π follows from Birkhoff's Theorem. Given such a permutation Π, the algorithm updates A to $A - \alpha\Pi$ where $\alpha = \min_i A_{i,\Pi(i)}$. The updated matrix A has non-negative entries and has strictly more zeros than the original A. Since the update decreases each row and

2. The decomposition is usually not unique and the implementation may have a bias as to exactly which convex combination is chosen.

Algorithm 1 PermELearn: Selecting a permutation

Require: a doubly stochastic $n \times n$ matrix W

$A := W$; $q = 0$;

repeat

$q := q + 1$;

Find permutation Π^q such that $A_{i,\Pi^q(i)}$ is positive for each $i \in \{1,\ldots,n\}$

$\alpha_q := \min_i A_{i,\Pi^q(i)}$

$A := A - \alpha_q \Pi^q$

until All entries of A are zero $\quad \{$at end of loop $W = \sum_{k=1}^{q} \alpha_k \Pi^k\}$

Randomly select and return a $\widehat{\Pi} \in \{\Pi^1,\ldots,\Pi^q\}$ using probabilities $\alpha_1,\ldots,\alpha_q$.

Algorithm 2 PermELearn: Weight Matrix Update

Require: learning rate η, loss matrix L, and doubly stochastic weight matrix W

Create W' where each $W'_{i,j} = W_{i,j} e^{-\eta L_{i,j}}$ (3)

Create doubly stochastic $\widetilde{W}$ by re-balancing the rows and columns of W' (Sinkhorn balancing) and update W to $\widetilde{W}$.

column sum by α and the original matrix W was doubly stochastic, each matrix A will have rows and columns that sum to the same amount. In other words, each matrix A created during Algorithm 1 is a constant times a doubly stochastic matrix, and thus (by Birkhoff's Theorem) is a constant times a convex combination of permutations.

After at most $n^2 - n$ iterations the algorithm arrives at a matrix A having exactly n non-zero entries, so this A is a constant times a permutation matrix. Therefore, Algorithm 1 decomposes the original doubly stochastic matrix into the convex combination of (at most) $n^2 - n + 1$ permutation matrices. The more refined arguments in Appendix A shows that the Algorithm 1 never uses more than $n^2 - 2n + 2$ permutations, matching the bound given by Birkhoff's Theorem.

Several improvements are possible. In particular, we need not compute each perfect matching from scratch. If only z entries of A are zeroed by a permutation, then that permutation is still a matching of size $n - z$ in the graph for the updated matrix. Thus we need to find only z augmenting paths to complete the perfect matching. The entire process thus requires finding $O(n^2)$ augmenting paths at a cost of $O(n^2)$ each, for a total cost of $O(n^4)$ to decompose weight matrix W into a convex combination of permutations.

4.1 Updating the Weights

In the second step, Algorithm 2 updates the weight matrix by multiplying each $W_{i,j}$ entry by the factor $e^{-\eta L_{i,j}}$. These factors destroy the row and column normalization, so the matrix must be re-balanced to restore the doubly-stochastic property. There is no closed form for the normalization step. The standard iterative re-balancing method for non-negative matrices is called *Sinkhorn balancing*. This method first normalizes each row of the matrix to sum to one, and then normalizes the columns. Since normalizing the columns typically destroys the row normalization, the process must be iterated until convergence (Sinkhorn, 1964).

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 1 \end{pmatrix} \xrightarrow{\text{Sinkhorn balancing}} \begin{pmatrix} \frac{\sqrt{2}}{1+\sqrt{2}} & \frac{1}{1+\sqrt{2}} \\ \frac{1}{1+\sqrt{2}} & \frac{\sqrt{2}}{1+\sqrt{2}} \end{pmatrix}$$

Figure 2: Example where Sinkhorn balancing requires infinitely many steps.

Normalizing the rows corresponds to pre-multiplying by a diagonal matrix. The product of these diagonal matrices thus represents the combined effect of the multiple row normalization steps. Similarly, the combined effect of the column normalization steps can be represented by post-multiplying the matrix by a diagonal matrix. Therefore we get the well known fact that Sinkhorn balancing a matrix A results in a doubly stochastic matrix RAC where R and C are diagonal matrices. Each entry $R_{i,i}$ is the positive multiplier applied to row i, and each entry $C_{j,j}$ is the positive multiplier of column j needed to convert A into a doubly stochastic matrix.

In Figure 2 we give a rational matrix that balances to an irrational matrix. Since each row and column balancing step creates rationals, Sinkhorn balancing produces irrationals only in the limit (after infinitely many steps). Multiplying a weight matrix from the left and/or right by non-negative diagonal matrices (e.g., row or column normalization) preserves the ratio of product weights between permutations. That is if $A' = RAC$, then for any two permutations Π_1 and Π_2,

$$\frac{\prod_i A'_{i,\Pi_1(i)}}{\prod_i A'_{i,\Pi_2(i)}} = \frac{\prod_i A_{i,\Pi_1(i)} R_{i,i} C_{\Pi_1(i),\Pi_1(i)}}{\prod_i A_{i,\Pi_2(i)} R_{i,i} C_{\Pi_2(i),\Pi_2(i)}} = \frac{\prod_i A_{i,\Pi_1(i)}}{\prod_i A_{i,\Pi_2(i)}}.$$

Therefore $\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1 \end{pmatrix}$ must balance to a doubly stochastic matrix $\begin{pmatrix} a & 1-a \\ 1-a & a \end{pmatrix}$ such that the ratio of the product weight between the two permutations $(1,2)$ and $(2,1)$ is preserved. This means $\frac{1/2}{1/4} = \frac{a^2}{(1-a)^2}$ and thus $a = \frac{\sqrt{2}}{1+\sqrt{2}}$.

This example leads to another important observation: PermELearn's predictions are different than Hedge's when each permutation is treated as an expert. If each permutation is explicitly represented as an expert, then the Hedge algorithm predicts permutation Π with probability proportional to the product weight, $\prod_i e^{-\eta \sum_t L^t_{i,\Pi(i)}}$. However, algorithm PermELearn predicts differently. With the weight matrix in Figure 4.1, Hedge puts probability $\frac{2}{3}$ on permutation $(1,2)$ and probability $\frac{1}{3}$ on permutation $(2,1)$ while PermELearn puts probability $\frac{\sqrt{2}}{1+\sqrt{2}} \approx 0.59$ on permutation $(1,2)$ and probability $\frac{\sqrt{1}}{1+\sqrt{2}} \approx 0.41$ on permutation (2,1).

There has been much written on the balancing of matrices, and we briefly describe only a few of the results here. Sinkhorn showed that this procedure converges and that the RAC balancing of any matrix A into a doubly stochastic matrix is unique (up to canceling multiples of R and C) if it exists[3] (Sinkhorn, 1964).

A number of authors consider balancing a matrix A so that the row and column sums are $1 \pm \varepsilon$. Franklin and Lorenz (1989) show that $O(\text{length}(A)/\varepsilon)$ Sinkhorn iterations suffice, where length(A) is the bit-length of matrix A's binary representation. Kalantari and Khachiyan (1996) show that

3. Some non-negative matrices, like $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$, cannot be converted into doubly stochastic matrices because of their pattern of zeros. The weight matrices we deal with have strictly positive entries, and thus can always be made doubly stochastic with an RAC balancing.

$O(n^4 \ln \frac{n}{\varepsilon} \ln \frac{1}{\min A_{i,j}})$ operations suffice using an interior point method. Linial et al. (2000) give a preprocessing step after which only $O((n/\varepsilon)^2)$ Sinkhorn iterations suffice. They also present a strongly polynomial time iterative procedure requiring $\tilde{O}(n^7 \log(1/\varepsilon))$ iterations. Balakrishnan et al. (2004) give an interior point method with complexity $O(n^6 \log(n/\varepsilon))$. Finally, Fürer (2004) shows that if the row and column sums of A are $1 \pm \varepsilon$ then every matrix entry changes by at most $\pm n\varepsilon$ when A is balanced to a doubly stochastic matrix.

4.2 Dealing with Approximate Balancing

With slight modifications, Algorithm PermELearn can handle the situation where its weight matrix is imperfectly balanced (and thus not quite doubly stochastic). As before, let W be the fully balanced doubly stochastic weight matrix, but we now assume that only an approximately balanced $\widehat{W}$ is available to predict from. In particular, we assume that each row and column of $\widehat{W}$ sum to $1 \pm \varepsilon$ for some $\varepsilon < \frac{1}{3}$. Let $s \geq 1 - \varepsilon$ be the smallest row or column sum in $\widehat{W}$.

We modify Algorithm 1 in two ways. First, A is initialized to $\frac{1}{s}\widehat{W}$ rather than W. This ensures every row and column in the initial A sums to at least one, to at most $1 + 3\varepsilon$, and at least one row or column sums to exactly 1. Second, the loop exits as soon as A has an all-zero row or column. Since the smallest row or column sum starts at 1, is decreased by α_k each iteration k, and ends at zero, we have that $\sum_{k=1}^{q} \alpha_k = 1$ and the modified Algorithm 1 still outputs a convex combination of permutations $C = \sum_{k=1}^{q} \alpha_k \Pi^k$. Furthermore, each entry $C_{i,j} \leq \frac{1}{s}\widehat{W}_{i,j}$. We now bound the additional loss of this modified algorithm.

Lemma 1 *If the weight matrix $\widehat{W}$ is approximately balanced so each row and column sum is in $1 \pm \varepsilon$ (for $\varepsilon \leq \frac{1}{3}$) then the modified Algorithm 1 has an expected loss $C \bullet L$ at most $3n^3\varepsilon$ greater than the expected loss $W \bullet L$ of the original algorithm that uses the completely balanced doubly stochastic matrix W.*

Proof Let s be the smallest row or column sum in $\widehat{W}$. Since each row and column sum of $\frac{1}{s}\widehat{W}$ lies in $[1, 1+3\varepsilon]$, each entry of $\frac{1}{s}\widehat{W}$ is close to the corresponding entry of the fully balanced W. In particular each $\frac{1}{s}\widehat{W}_{i,j} \leq W_{i,j} + 3n\varepsilon$ (Fürer, 2004). This allows us to bound the expected loss when predicting with the convex combination C in terms of the expected loss using a decomposition of the perfectly balanced W:

$$\begin{aligned} C \bullet L &\leq \frac{1}{s}\widehat{W} \bullet L \\ &= \sum_{i,j} \frac{\widehat{W}_{i,j}}{s} L_{i,j} \\ &\leq \sum_{i,j} (W_{i,j} + 3n\varepsilon) L_{i,j} \\ &\leq W \bullet L + 3n^3\varepsilon. \end{aligned}$$

Therefore the extra loss incurred by using a ε-approximately balanced weight matrix at a particular trial is at most $3n^3\varepsilon$, as desired. ■

If in a sequence of T trials the matrices $\widehat{W}$ are $\varepsilon = 1/(3Tn^3)$ balanced (so that each row and column sum is $1 \pm 1/(3Tn^3)$) then Lemma 1 implies that the total additional expected loss for using approximate balancing is at most 1. The algorithm of Balakrishnan et al. (2004) ε-balances a matrix in $O(n^6 \log(n/\varepsilon))$ time (note that this dominates the time for the loss update and constructing the convex combination). This balancing algorithm with $\varepsilon = 1/(3Tn^3)$ together with the modified prediction algorithm give a method requiring $O(Tn^6 \log(Tn))$ total time over the T trials and having a bound of $\sqrt{2\mathcal{L}_{\text{est}} n \ln n} + n \ln n + 1$ on the worst-case regret.

If the number of trials T is not known in advance then setting ε as a function of t can be helpful. A natural choice is $\varepsilon_t = 1/(3t^2n^3)$. In this case the total extra regret for not having perfect balancing is bounded by $\sum_{t=1}^{T} 1/t^2 \leq 5/3$ and the total computation time over the T trials is still bounded by $O(Tn^6 \log(Tn))$.

One might be concerned about the effects of approximate balancing propagating between trials. However this is not an issue. In the following section we show that the loss updates and balancing can be arbitrarily interleaved. Therefore the modified algorithm can either keep a cumulative loss matrix $L^{\leq t} = \sum_{i=1}^{t} L^i$ and create its next $\widehat{W}$ by (approximately) balancing the matrix with entries $\frac{1}{n} e^{-\eta L_{i,j}^{\leq t}}$, or apply the multiplicative updates to the previous approximately balanced $\widehat{W}$.

5. Bounds for PermELearn

Our analysis of PermELearn follows the entropy-based analysis of the exponentiated gradient family of algorithms (Kivinen and Warmuth, 1997). This style of analysis first shows a per-trial progress bound using relative entropy to a comparator as a measure of progress, and then sums this invariant over the trials to bound the expected total loss of the algorithm. We also show that PermELearn's weight update belongs to the exponentiated gradient family of updates (Kivinen and Warmuth, 1997) since it is the solution to a minimization problem that trades of the loss (in this case a linear loss) against a relative entropy regularization.

Recall that the expected loss of PermELearn on a trial is a linear function of its weight matrix W. Therefore the gradient of the loss is independent of the current value of W. This property of the loss greatly simplifies the analysis. Our analysis for this setting provides a good foundation for learning permutation matrices and lays the groundwork for the future study of other permutation loss functions.

We start our analysis with an attempt to mimic the standard analysis (Kivinen and Warmuth, 1997) for the exponentiated gradient family updates which multiply by exponential factors and re-normalize. The per-trial invariant used to analyze the exponentiated gradient family bounds the decrease in relative entropy from any (normalized) vector u to the algorithm's weight vector by a linear combination of the algorithm's loss and the loss of u on the trial. In our case the weight vectors are matrices and we use the following (un-normalized) relative entropy between matrices A and B with non-negative entries:

$$\Delta(A,B) = \sum_{i,j} A_{i,j} \ln \frac{A_{i,j}}{B_{i,j}} + B_{i,j} - A_{i,j} \ .$$

Note that this is just the sum of the relative entropies between the corresponding rows (or equivalently, between the corresponding columns):

$$\Delta(A,B) = \sum_{i} \Delta(A_{i,\star}, B_{i,\star}) = \sum_{j} \Delta(A_{\star,j}, B_{\star,j})$$

(here $A_{i,\star}$ is the ith row of A and $A_{\star,j}$ is its jth column).

Unfortunately, the lack of a closed form for the matrix balancing procedure makes it difficult to prove bounds on the loss of the algorithm. Our solution is to break PermELearn's update (Algorithm 2) into two steps, and use only the progress made to the intermediate un-balanced matrix in our per-trial bound (8). After showing that balancing to a doubly stochastic matrix only increases the progress, we can sum the per-trial bound to obtain our main theorem.

5.1 A Dead End

In each trial, PermELearn multiplies each entry of its weight matrix by an exponential factor and then uses one additional factor per row and column to make the matrix doubly stochastic (Algorithm 2 described in Section 4.1):

$$\widetilde{W}_{i,j} := r_i c_j W_{i,j} e^{-\eta L_{i,j}} \tag{4}$$

where the r_i and c_j factors are chosen so that all rows and columns of the matrix $\widetilde{W}$ sum to one.

We now show that PermELearn's update (4) gives the matrix A solving the following minimization problem:

$$\begin{array}{l} \operatorname{argmin} \\ \forall i : \sum_j A_{i,j} = 1 \\ \forall j : \sum_i A_{i,j} = 1 \end{array} \quad (\Delta(A,W) + \eta\,(A \bullet L)). \tag{5}$$

Since the linear constraints are feasible and the divergence is strictly convex, there always is a unique solution, even though the solution does not have a closed form.

Lemma 2 *PermELearn's updated weight matrix* $\widetilde{W}$ (4) *is the solution of* (5).

Proof We form a Lagrangian for the optimization problem:

$$l(A,\rho,\gamma) = \Delta(A,W) + \eta\,(A \bullet L) + \sum_i \rho_i (\sum_j A_{i,j} - 1) + \sum_j \gamma_j (\sum_i A_{i,j} - 1).$$

Setting the derivative with respect to $A_{i,j}$ to 0 yields $A_{i,j} = W_{i,j} e^{-\eta L_{i,j}} e^{-\rho_i} e^{-\gamma_j}$. By enforcing the row and column sum constraints we see that the factors $r_i = e^{-\rho_i}$ and $c_j = e^{-\gamma_j}$ function as row and column normalizers, respectively. ■

We now examine the progress $\Delta(U,W) - \Delta(U,\widetilde{W})$ towards an arbitrary stochastic matrix U. Using Equation (4) and noting that all three matrices are doubly stochastic (so their entries sum to n), we see that

$$\Delta(U,W) - \Delta(U,\widetilde{W}) = -\eta\, U \bullet L + \sum_i \ln r_i + \sum_j \ln c_j.$$

Making this a useful invariant requires lower bounding the sums on the rhs by a constant times $W \bullet L$, the loss of the algorithm. Unfortunately we are stuck because the r_i and c_j normalization factors don't even have a closed form.

5.2 Successful Analysis

Our successful analysis splits the update (4) into two steps:

$$W'_{i,j} := W_{i,j} e^{-\eta L_{i,j}} \quad \text{and} \quad \widetilde{W}_{i,j} := r_i c_j W'_{i,j}, \tag{6}$$

where (as before) r_i and c_j are chosen so that each row and column of the matrix $\widetilde{W}$ sum to one. Using the Lagrangian (as in the proof of Lemma 2), it is easy to see that these W' and $\widetilde{W}$ matrices solve the following minimization problems:

$$W' = \operatorname*{argmin}_A \left(\Delta(A,W) + \eta\,(A \bullet L)\right) \quad \text{and} \quad \widetilde{W} := \operatorname*{argmin}_{\substack{\forall i: \sum_j A_{i,j} = 1 \\ \forall j: \sum_i A_{i,j} = 1}} \Delta(A,W'). \tag{7}$$

The second problem shows that the doubly stochastic matrix $\widetilde{W}$ is the projection of W' onto to the linear row and column sum constraints. The strict convexity of the relative entropy between non-negative matrices and the feasibility of the linear constraints ensure that the solutions for both steps are unique.

We now lower bound the progress $\Delta(U,W) - \Delta(U,W')$ in the following lemma to get our per-trial invariant.

Lemma 3 *For any $\eta > 0$, any doubly stochastic matrices U and W and any trial with loss matrix $L \in [0,1]^{n \times n}$*

$$\Delta(U,W) - \Delta(U,W') \geq (1 - e^{-\eta})(W \bullet L) - \eta(U \bullet L),$$

where W' is the unbalanced intermediate matrix (6) *constructed by PermELearn from W.*

Proof The proof manipulates the difference of relative entropies and uses the inequality $e^{-\eta x} \leq 1 - (1 - e^{-\eta})x$, which holds for any η and any $x \in [0,1]$:

$$\begin{aligned}
\Delta(U,W) - \Delta(U,W') &= \sum_{i,j} \left(U_{i,j} \ln \frac{W'_{i,j}}{W_{i,j}} + W_{i,j} - W'_{i,j} \right) \\
&= \sum_{i,j} \left(U_{i,j} \ln(e^{-\eta L_{i,j}}) + W_{i,j} - W_{i,j} e^{-\eta L_{i,j}} \right) \\
&\geq \sum_{i,j} \left(-\eta L_{i,j} U_{i,j} + W_{i,j} - W_{i,j}(1 - (1 - e^{-\eta}) L_{i,j}) \right) \\
&= -\eta (U \bullet L) + (1 - e^{-\eta})(W \bullet L).
\end{aligned}$$

■

Relative entropy is a Bregman divergence, so the Generalized Pythagorean Theorem (Bregman, 1967) applies. Specialized to our setting, this theorem states that if S is a closed convex set containing some matrix U with non-negative entries, W' is any matrix with strictly positive entries, and $\widetilde{W}$ is the relative entropy projection of W' onto S then

$$\Delta(U,W') \geq \Delta(U,\widetilde{W}) + \Delta(\widetilde{W},W').$$

Furthermore, this holds with equality when S is affine, which is the case here since S is the set of matrices whose rows and columns each sum to 1. Rearranging and noting that $\Delta(A,B)$ is non-negative yields Corollary 3 of Herbster and Warmuth (2001), which is the inequality we need:

$$\Delta(U,W') - \Delta(U,\widetilde{W}) = \Delta(\widetilde{W},W') \geq 0.$$

Combining this with the inequality of Lemma 3 gives the critical per-trial invariant:

$$\Delta(U,W) - \Delta(U,\widetilde{W}) \geq (1-e^{-\eta})(W \bullet L) - \eta(U \bullet L). \tag{8}$$

We now introduce some notation and bound the expected total loss by summing the above inequality over a sequence of trials. When considering a sequence of trials, L^t is the loss matrix at trial t, W^{t-1} is PermELearn's weight matrix W at the start of trial t (so W^0 is the initial weight matrix) and W^t is the updated weight matrix $\widetilde{W}$ at the end of the trial.

Theorem 4 *For any learning rate $\eta > 0$, any doubly stochastic matrices U and initial W^0, and any sequence of T trials with loss matrices $L^t \in [0,1]^{n\times n}$ (for $1 \leq t \leq T$), the expected loss of PermELearn is bounded by:*

$$\sum_{t=1}^{T} W^{t-1} \bullet L^t \leq \frac{\Delta(U,W^0) - \Delta(U,W^T) + \eta \sum_{t=1}^{T} U \bullet L^t}{1-e^{-\eta}} \ .$$

Proof Applying (8) to trial t gives:

$$\Delta(U,W^{t-1}) - \Delta(U,W^t) \geq (1-e^{-\eta})(W^{t-1} \bullet L^t) - \eta(U \bullet L^t).$$

By summing the above over all T trials we get:

$$\Delta(U,W^0) - \Delta(U,W^T) \geq (1-e^{-\eta}) \sum_{t=1}^{T} W^{t-1} \bullet L^t - \eta \sum_{t=1}^{T} U \bullet L^t.$$

The bound then follows by solving for the total expected loss, $\sum_{t=1}^{T} W^{t-1} \bullet L^t$, of the algorithm. ■

When the entries of W^0 are all initialized to $\frac{1}{n}$ and U is a permutation then $\Delta(U,W^0) = n\ln n$. Since each doubly stochastic matrix U is a convex combination of permutation matrices, at least one minimizer of the total loss $\sum_{t=1}^{T} U \bullet L$ will be a permutation matrix. If $\mathcal{L}_{\text{best}}$ denotes the loss of such a permutation U^*, then Theorem 4 implies that the total loss of the algorithm is bounded by

$$\frac{\Delta(U^*,W^0) + \eta \mathcal{L}_{\text{best}}}{1-e^{-\eta}}.$$

If upper bounds $\Delta(U^*,W^0) \leq D_{\text{est}} \leq n\ln n$ and $\mathcal{L}_{\text{est}} \geq \mathcal{L}_{\text{best}}$ are known, then by choosing $\eta = \ln\left(1+\sqrt{\frac{2D_{\text{est}}}{\mathcal{L}_{\text{est}}}}\right)$, and the above bound becomes (Freund and Schapire, 1997):

$$\mathcal{L}_{\text{best}} + \sqrt{2\mathcal{L}_{\text{est}} D_{\text{est}}} + \Delta(U^*,W^0). \tag{9}$$

A natural choice for D_{est} is $n\ln n$. In this case the tuned bound becomes

$$\mathcal{L}_{\text{best}} + \sqrt{2\mathcal{L}_{\text{est}} n\ln n} + n\ln n.$$

5.3 Approximate Balancing

The preceding analysis assumes that PermELearn's weight matrix is perfectly balanced each iteration. However, balancing techniques are only capably of approximately balancing the weight matrix in finite time, so implementations of PermELearn must handle approximately balanced matrices. In Section 4.2, we describe an implementation that uses an approximately balanced $\widehat{W}^{t-1}$ at the start of iteration t rather than the completely balanced W^{t-1} of the preceding analysis. Lemma 1 shows that when this implementation of PermELearn uses an approximately balanced $\widehat{W}^{t-1}$ where each row and column sum is in $1 \pm \varepsilon_t$, then the expected loss on trial t is at most $W^{t-1} \bullet L^t + 3n^3\varepsilon_t$. Summing over all trials and using Theorem 4, this implementation's total loss is at most

$$\sum_{t=1}^{T} \left(W^{t-1} \bullet L^t + 3n^3\varepsilon_t\right) \leq \frac{\Delta(U,W^0) - \Delta(U,W^T) + \eta \sum_{t=1}^{T} U \bullet L^t}{1-e^{-\eta}} + \sum_{t=1}^{T} 3n^3\varepsilon_t \ .$$

As discussed in Section 4.2, setting $\varepsilon_t = 1/(3n^3t^2)$ leads to an additional loss of less than $5/3$ over the bound of Theorem 4 and its subsequent tunings while incurring a total running time (over all T trials) in $O(Tn^6 \log(Tn))$. In fact, the additional loss for approximate balancing can be made less than any positive c by setting $\varepsilon_t = c/(5n^3t^2)$. Since the time to approximately balance depends only logarithmically on $1/\varepsilon$, the total time taken over T trials remains in $O(Tn^6 \log(Tn))$.

5.4 Split Analysis for the Hedge Algorithm

Perhaps the simplest case where the loss is linear in the parameter vector is the on-line allocation setting of Freund and Schapire (1997). It is instructive to apply our method of splitting the update in this simpler setting. There are N experts and the algorithm keeps a probability distribution w over the experts. In each trial the algorithm picks expert i with probability w_i and then gets a loss vector $\ell \in [0,1]^N$. Each expert i incurs loss ℓ_i and the algorithm's expected loss is $w \cdot \ell$. Finally w is updated to $\widetilde{w}$ for the next trial.

The Hedge algorithm (Freund and Schapire, 1997) updates its weight vector to $\tilde{w}_i = \frac{w_i e^{-\eta \ell_i}}{\sum_j w_j e^{-\eta \ell_j}}$. This update can be motivated by a tradeoff between the un-normalized relative entropy to the old weight vector and expected loss in the last trial (Kivinen and Warmuth, 1999):

$$\widetilde{w} := \operatorname*{argmin}_{\sum_i \widehat{w}_i = 1} \left(\Delta(\widehat{w}, w) + \eta \ \widehat{w} \cdot \ell\right).$$

For vectors, the relative entropy is simply $\Delta(\widehat{w}, w) := \sum_i \widehat{w}_i \ln \frac{\widehat{w}_i}{w_i} + w_i - \widehat{w}_i$. As in the permutation case, we can split this update (and motivation) into two steps: setting each $w'_i = w_i e^{-\eta \ell_i}$ then $\widetilde{w} = w' / \sum_i w'_i$. These are the solutions to:

$$w' := \operatorname*{argmin}_{\widehat{w}} \left(\Delta(\widehat{w}, w) + \eta \ \widehat{w} \cdot \ell\right) \ \text{ and } \ \widetilde{w} := \operatorname*{argmin}_{\sum_i \widehat{w}_i = 1} \Delta(\widehat{w}, w').$$

The following lower bound has been shown on the progress towards any probability vector u serving as a comparator:[4]

$$\begin{aligned}\Delta(u,w)-\Delta(u,\widetilde{w}) &= -\eta\, u\cdot\ell-\ln\sum_i w_i e^{-\eta\ell_i}\\ &\geq -\eta\, u\cdot\ell-\ln\sum_i w_i(1-(1-e^{-\eta})\ell_i)\\ &\geq -\eta\, u\cdot\ell+w\cdot\ell\,(1-e^{-\eta})\,, \end{aligned}\tag{10}$$

where the first inequality uses $e^{-\eta x}\leq 1-(1-e^{-\eta})x$, for any $x\in[0,1]$, and the second uses $-\ln(1-x)\geq x$, for $x\in[0,1]$. Surprisingly the same inequality already holds for the un-normalized update:[5]

$$\Delta(u,w)-\Delta(u,w')=-\eta\, u\cdot\ell+\sum_i w_i(1-e^{-\eta\ell_i})\geq w\cdot\ell\,(1-e^{-\eta})-\eta\, u\cdot\ell.$$

Since the normalization is a projection w.r.t. a Bregman divergence onto a linear constraint satisfied by the comparator u, $\Delta(u,w')-\Delta(u,\widetilde{w})\geq 0$ by the Generalized Pythagorean Theorem (Herbster and Warmuth, 2001). The total progress for both steps is again Inequality (10).

With the key Inequality (10) in hand, it is easy to introduce trial dependent notation and sum over trails (as done in the proof of Theorem 4, arriving at the familiar bound for Hedge (Freund and Schapire, 1997): For any $\eta>0$, any probability vectors w^0 and u, and any loss vectors $\ell^t\in[0,1]^n$,

$$\sum_{t=1}^{T} w^{t-1}\bullet\ell^t\leq\frac{\Delta(u,w^0)-\Delta(u,w^T)+\eta\sum_{t=1}^{T}u\bullet\ell^t}{1-e^{-\eta}}\quad.\tag{11}$$

Note that the r.h.s. is actually constant in the comparator u (Kivinen and Warmuth, 1999), that is, for all u,

$$\frac{\Delta(u,w^0)-\Delta(u,w^T)+\eta\sum_{t=1}^{T}u\bullet\ell^t}{1-e^{-\eta}}=\frac{-\ln\sum_i w_i^0 e^{-\eta\ell_i^{\leq T}}}{1-e^{\eta}}.$$

The r.h.s. of the above equality is often used as a potential in proving bounds for expert algorithms. We discuss this further in Appendix B.

5.5 When to Normalize?

Probably the most surprising aspect about the proof methodology is the flexibility about how and when to project onto the constraints. Instead of projecting a nonnegative matrix onto all $2n$ constraints at once (as in optimization problem (7)), we could mimic the Sinkhorn balancing algorithm by first projecting onto the row constraints and then the column constraints and alternating until convergence. The Generalized Pythagorean Theorem shows that projecting onto *any* convex constraint that is satisfied by the comparator class of doubly stochastic matrices brings the weight matrix closer to *every* doubly stochastic matrix.[6] Therefore our bound on $\sum_t W^{t-1}\bullet L^t$ (Theorem 4) holds if the exponential updates are interleaved with any sequence of projections to some subsets of the

4. This is essentially Lemma 5.2 of Littlestone and Warmuth (1994). The reformulation of this type of inequality with relative entropies goes back to Kivinen and Warmuth (1999)
5. Note that if the algorithm does not normalize the weights then w is no longer a distribution. When $\sum_i w_i<1$, the loss $w\cdot L$ amounts to incurring 0 loss with probability $1-\sum_i w_i$, and predicting as expert i with probability w_i.
6. There is a large body of work on finding a solution subject to constraints via iterated Bregman projections (see, e.g., Censor and Lent, 1981).

constraints. However, if the normalization constraints are not enforced then W is no longer a convex combination of permutations. Furthermore, the exponential update factors only decrease the entries of W and without any normalization all of the entries of W can get arbitrarily small. If this is allowed to happen then the "loss" $W \bullet L$ can approach 0 for any loss matrix, violating the spirit of the prediction model.

There is a direct argument that shows that the same final doubly stochastic matrix is reached if we interleave the exponential updates with projections to any of the constraints as long as all $2n$ constraints hold at the end. To see this we partition the class of matrices with positive entries into equivalence classes. Call two such matrices A and B *equivalent* if there are diagonal matrices R and C with positive diagonal entries such that $B = RAC$. Note that $[RAC]_{i,j} = R_{i,i}A_{i,j}C_{j,j}$ and therefore B is just a rescaled version of A. Projecting onto any row and/or column sum constraints amounts to pre- and/or post-multiplying the matrix by some positive diagonal matrices R and C. Therefore if matrices A and B are equivalent then the projection of A (or B) onto a set of row and/or column sum constraints results in another matrix equivalent to both A and B.

The importance of equivalent matrices is that they balance to the same doubly stochastic matrix.

Lemma 5 *For any two equivalent matrices A and RAC, where the entries of A and the diagonal entries of R and C are positive,*

$$\underset{\substack{\forall i: \sum_j \widehat{A}_{i,j} = 1 \\ \forall j: \sum_i \widehat{A}_{i,j} = 1}}{\operatorname{argmin}} \Delta(\widehat{A}, A) \quad = \quad \underset{\substack{\forall i: \sum_j \widehat{A}_{i,j} = 1 \\ \forall j: \sum_i \widehat{A}_{i,j} = 1}}{\operatorname{argmin}} \Delta(\widehat{A}, RAC).$$

Proof The strict convexity of the relative entropy implies that both problems have a unique matrix as their solution. We will now reason that the unique solutions for both problems are the same. By using a Lagrangian (as in the proof of Lemma 2) we see that the solution of the left optimization problem is a square matrix with $\dot{r}_i A_{i,j} \dot{c}_j$ in position i, j. Similarly the solution of the problem on the right has $\ddot{r}_i R_{i,i} A_{i,j} C_{j,j} \ddot{c}_j$ in position i, j. Here the factors $\dot{r}_i, \ddot{r}_i$ function as row normalizers and $\dot{c}_j, \ddot{c}_j$ as column normalizers. Given a solution matrix $\dot{r}_i, \dot{c}_j$ to the left problem, then $\dot{r}_i / R_{i,i}, \dot{c}_j / C_{j,j}$ is a solution of the right problem of the same value. Also if $\ddot{r}_i, \ddot{c}_j$ is a solution of right problem, then $\ddot{r}_i R_{i,i}, \ddot{c}_j C_{j,j}$ is a solution to the left problem of the same value.

This shows that both minimization problems have the same value and the matrix solutions for both problems are the same and unique (even though the normalization factors $\dot{r}_i, \dot{c}_j$ of say the left problem are not necessarily unique). Note that its crucial for the above argument that the diagonal entries of R, C are positive. ■

The analogous phenomenon is much simpler in the weighted majority case: Two non-negative vectors a and b are *equivalent* if $a = cb$, where c is any nonnegative scalar, and again each equivalence class has exactly one normalized weight vector.

PermELearn's intermediate matrix $W'_{i,j} := W_{i,j} e^{-\eta L_{i,j}}$ can be written $W \circ M$ where $\circ$ denotes the *Hadamard* (entry-wise) *Product* and $M_{i,j} = e^{-\eta L_{i,j}}$. Note that the Hadamard product commutes with matrix multiplication by diagonal matrices, if C is diagonal and $P = (A \circ B)C$ then $P_{i,j} = (A_{i,j} B_{i,j}) C_{j,j} = (A_{i,j} C_{j,j}) B_{i,j}$ so we also have $P = (AC) \circ B$. Similarly, $R(A \circ B) = (RA) \circ B$ when R is diagonal.

Hadamard products also preserve equivalence. For equivalent matrices A and $B = RAC$ (for diagonal R and C) the matrices $A \circ M$ and $B \circ M$ are equivalent (although they are not likely to be equivalent to A and B) since $B \circ M = (RAC) \circ M = R(A \circ M)C$.

This means that any two runs of PermELearn-like algorithms that have the same bag of loss matrices and equivalent initial matrices end with equivalent final matrices even if they project onto different subsets of the constraints at the end of the various trials.

In summary the proof method discussed so far uses a relative entropy as a measure of progress and relies on Bregman projections as its fundamental tool. In Appendix B we re-derive the bound for PermELearn using the value of the optimization problem (5) as a potential. This value is expressed using the dual optimization problem and intuitively the application of the Generalized Pythagorean Theorem now is replaced by plugging in a non-optimal choice for the dual variables. Both proof techniques are useful.

5.6 Learning Mappings

We have an algorithm that has small regret against the best permutation. Permutations are a subset of all mappings from $\{1,\ldots,n\}$ to $\{1,\ldots,n\}$. We continue using Π for a permutation and introduce Ψ to denote an arbitrary mapping from $\{1,\ldots,n\}$ to $\{1,\ldots,n\}$. Mappings differ from permutations in that the n dimensional vector $(\Psi(i))_{i=1}^n$ can have repeats, that is, $\Psi(i)$ might equal $\Psi(j)$ for $i \neq j$. Again we alternately represent a mapping Ψ as an $n \times n$ matrix where $\Psi_{i,j} = 1$ if $\Psi(i) = j$ and 0 otherwise. Note that such square[7] *mapping matrices* have the special property that they have exactly one 1 in each row. Again the loss is specified by a loss matrix L and the loss of mapping Ψ is $\Psi \bullet L$.

It is straightforward to design an algorithm *MapELearn* for learning mappings with exponential weights: Simply run n independent copies of the Hedge algorithm for each of the n rows of the received loss matrices. That is, the r'th copy of Hedge always receives the r'th row of the loss matrix L as its loss vector. Even though learning mappings is easy, it is nevertheless instructive to discuss the differences with PermELearn.

Note that MapELearn's combined weight matrix is now a convex combination of mappings, that is, a "singly" stochastic matrix with the constraint that each row sums to one. Again, after the exponential update (3), the constraints are typically not satisfied any more, but they can be easily reestablished by simply normalizing each row. The row normalization only needs to be done once in each trial: no iterative process is needed. Furthermore, no fancy decomposition algorithm is needed in MapELearn: for (singly) stochastic weight matrix W, the prediction $\Psi(i)$ is simply a random element chosen from the row distribution $W_{i,*}$. This sampling procedure produces a mapping Ψ such that $W = \mathbb{E}(\Psi)$ and thus $\mathbb{E}(\Psi \bullet L) = W \bullet L$ as needed.

We can use the same relative entropy between the single stochastic matrices, and the lower bound on the progress for the exponential update given in Lemma 3 still holds. Also our main bound (Theorem 4) is still true for MapELearn and we arrive at the same tuned bound for the total loss of MapELearn:

$$\mathcal{L}_{\text{best}} + \sqrt{2\mathcal{L}_{\text{est}} D_{\text{est}}} + \Delta(U^*, W^0),$$

where $\mathcal{L}_{\text{best}}$, $\mathcal{L}_{\text{est}}$, and D_{est} are now the total loss of the best mapping, a known upper bound on $\mathcal{L}_{\text{best}}$, and an upper bound on $\Delta(U^*, W^0)$, respectively. Recall that $\mathcal{L}_{\text{est}}$ and D_{est} are needed to tune the η parameter.

7. In the case of mappings the restriction to square matrices is not essential.

Our algorithm PermElearn for permutations may be seen as the above algorithm for mappings while enforcing the column sum constraints in addition to the row constraints used in MapELearn. Since PermELearn's row balancing "messes up" the column sums and vice versa, an interactive procedure (i.e., Sinkhorn Balancing) is needed to create to a matrix in which each row *and* column sums to one. The enforcement of the additional column sum constraints results in a doubly stochastic matrix, an apparently necessary step to produce predictions that are permutations (and an expected prediction equal to the doubly stochastic weight matrix).

When it is known that the comparator is a permutation, then the algorithm always benefits from enforcing the additional column constraints. In general we should always make use of any constraints that the comparator is known to satisfy (see, e.g., Warmuth and Vishwanathan, 2005, for a discussion of this).

As discussed in Section 4.1, if A' is a Sinkhorn-balanced version of a non-negative matrix A, then

$$\text{for any permutations } \Pi_1 \text{ and } \Pi_2, \quad \frac{\prod_i A_{i,\Pi_1(i)}}{\prod_i A_{i,\Pi_2(i)}} = \frac{\prod_i A'_{i,\Pi_1(i)}}{\prod_i A'_{i,\Pi_2(i)}}. \tag{12}$$

An analogous invariant holds for mappings: If A' is a row-balanced version of a non-negative matrix A, then

$$\text{for any mappings } \Psi_1 \text{ and } \Psi_2, \quad \frac{\prod_i A_{i,\Psi_1(i)}}{\prod_i A_{i,\Psi_2(i)}} = \frac{\prod_i A'_{i,\Psi_1(i)}}{\prod_i A'_{i,\Psi_2(i)}}.$$

However it is important to note that column balancing does not preserve the above invariant for mappings. In fact, permutations are the subclass of mappings where invariant 12 holds.

There is another important difference between PermELearn and MapELearn. For MapELearn, the probability of predicting mapping Ψ with weight matrix W is always the product $\prod_i W_{i,\Psi(i)}$. The analogous property does *not* hold for PermELearn. Consider the balanced 2×2 weight matrix W on the right of Figure 2. This matrix decomposes into $\frac{\sqrt{2}}{1+\sqrt{2}}$ times the permutation $(1,2)$ plus $\frac{1}{1+\sqrt{2}}$ times the permutation $(2,1)$. Thus the probability of predicting with permutation $(1,2)$ is $\sqrt{2}$ times the probability of permutation $(2,1)$ for the PermELearn algorithm. However, when the probabilities are proportional to the intuitive product form $\prod_i W_{i,\Pi(i)}$, then the probability ratio for these two permutations is 2. Notice that this intuitive product weight measure is the distribution used by the Hedge algorithm that explicitly treats each permutation as a separate expert. Therefore PermELearn is clearly different than a concise implementation of Hedge for permutations.

6. Follow the Perturbed Leader Algorithm

Perhaps the simplest on-line algorithm is the *Follow the Leader* (FL) algorithm: at each trial predict with one of the best models on the data seen so far. Thus FL predicts at trial t with an expert in $\operatorname{argmin}_i \ell_i^{<t}$ or any permutation in $\operatorname{argmin}_\Pi \Pi \bullet L^{<t}$, where "$<t$" indicates that we sum over the past trials, that is, $\ell_i^{<t} := \sum_{q=1}^{t-1} \ell_i^q$. The FL algorithm is clearly non-optimal; in the expert setting there is a simple adversary strategy that forces FL to have loss at least n times larger than the loss of the best expert in hindsight.

The expected total loss of tuned Hedge is one times the loss of the best expert plus lower order terms. Hedge achieves this by randomly choosing experts. The probability w_i^{t-1} for choosing expert i at trial t is proportional to $e^{-\eta \ell_i^{<t}}$. As the learning rate $\eta \to \infty$, Hedge becomes FL (when there are

no ties) and the same holds for PermELearn. Thus the exponential weights with moderate η may be seen as a soft min calculation: the algorithm hedges its bets and does not put all its probability on the expert with minimum loss so far.

The "Follow the Perturbed Leader" (FPL) algorithm of Kalai and Vempala (2005) is an alternate on-line prediction algorithm that works in a very general setting. It adds random perturbations to the total losses of the experts incurred so far and then predicts with the expert of minimum perturbed loss. Their FPL* algorithm has bounds closely related to Hedge and other multiplicative weight algorithms and in some cases Hedge can be simulated exactly (Kuzmin and Warmuth, 2005) by judiciously choosing the distribution of perturbations. However, for the permutation problem the bounds we were able to obtain for FPL* are weaker than the the bound we obtained bounds for PermELearn that uses exponential weights despite the apparent similarity between our representations and the general formulation of FPL*.

The FPL setting uses an abstract k-dimensional decision space used to encode predictors as well as a k-dimensional state space used to represent the losses of the predictors. At any trial, the current loss of a particular predictor is the dot product between that predictor's representation in the decision space and the state-space vector for the trial. This general setting can explicitly represent each permutation and its loss when $k = n!$. The FPL setting also easily handles the encodings of permutations and losses used by PermELearn by representing each permutation matrix Π and loss matrix L as n^2-dimensional vectors.

The FPL* algorithm (Kalai and Vempala, 2005) takes a parameter ε and maintains a cumulative loss matrix C (initially C is the zero matrix) At each trial, FPL*:

1. Generates a random perturbation matrix P where each $P_{i,j}$ is proportional to $\pm r_{i,j}$ where $r_{i,j}$ is drawn from the standard exponential distribution.

2. Predicts with a permutation Π minimizing $\Pi \bullet (C+P)$.

3. After getting the loss matrix L, updates C to $C+L$.

Note that FPL* is more computationally efficient than PermELearn. It takes only $O(n^3)$ time to make its prediction (the time to compute a minimum weight bipartite matching) and only $O(n^2)$ time to update C. Unfortunately the generic FPL* loss bounds are not as good as the bounds on PermELearn. In particular, they show that the loss of FPL* on any sequence of trials is at most[8]

$$(1+\varepsilon)\mathcal{L}_{\text{best}} + \frac{8n^3(1+\ln n)}{\varepsilon}$$

where ε is a parameter of the algorithm. When the loss of the best expert is known ahead of time, ε can be tuned and the bound becomes

$$\mathcal{L}_{\text{best}} + 4\sqrt{2\mathcal{L}_{\text{best}} n^3(1+\ln n)} + 8n^3(1+\ln n) \quad .$$

Although FPL* gets the same $\mathcal{L}_{\text{best}}$ leading term, the excess loss over the best permutation grows as $n^3 \ln n$ rather the $n \ln n$ growth of PermELearn's bound. Of course, PermELearn pays for the improved bound by requiring more computation.

8. The n^3 terms in the bounds for FPL are n times the sum of the entries in the loss matrix. So if the application has a loss motif whose entries sum to only n, then the n^3 factors become n^2.

It is important to note that Kalai and Vempala also present a refined analysis of FPL* when the perturbed leader changes only rarely. This analysis leads to bounds that are similar to the bounds given by the entropic analysis of the Hedge algorithm (although the constant on the square-root term is not quite as good). However, this refined analysis cannot be directly applied with the efficient representations of permutations because the total perturbations associated with different permutations are no longer independent exponentials. We leave the adaptation of the refined analysis to the permutation case as an open problem.

7. Lower Bounds

In this section we prove lower bounds on the worst-case regret of any algorithm for our permutation learning problem by reducing the expert allocation problem for n experts with loss range $[0,n]$ to the permutation learning problem. We then show in Appendix C a lower bound for this n expert allocation problem that uses a known lower bound in the expert advice setting with losses in $[0,1]$.

For the reduction we choose any set of n permutations $\{\Pi^1,\ldots,\Pi^n\}$ that use disjoint positions, that is, $\sum_{i=1}^n \Pi^i$ is the $n\times n$ matrix of all ones. Using disjoint positions ensures that the losses of these n permutations can be set independently. Each Π^i matrix in this set corresponds to the ith expert in the n-expert allocation problem. To simulate an n-expert trial with loss vector $\ell\in[0,n]^n$ we use a loss matrix L s.t. $\Pi^i\bullet L=\ell_i$. This is done by setting all entries in $\{L_{q,\Pi^i(q)} : 1\le q\le n\}$ to $\ell_i/n\in[0,1]$, that is, $L=\sum_i \Pi^i(\ell_i/n)$. Now for any doubly stochastic matrix W,

$$W\bullet L=\sum_i \frac{\Pi^i\bullet W}{n}\,\ell_i.$$

Note that the n dimensional vector with the components $(\Pi^i\bullet W)/n$ is a probability vector and therefore any algorithm for the n-element permutation problem can be used as an algorithm for the n-expert allocation problem with losses in the range $[0,n]$. Thus any lower bound for the latter model is also a lower bound on the n-element permutation problem.

We first prove a lower bound for the case when at least one expert has loss zero for the entire sequence of trials. If the algorithm allocates any weight to experts that have already incurred positive loss, then the adversary can assign loss only to those experts and force the algorithm increase its expected loss without reducing the number of experts of loss zero. Thus we can assume w.l.o.g. that the algorithm allocates positive weight only to experts of zero loss. The algorithm minimizes its expected loss and the adversary maximizes it. We get a lower bound by fixing the adversary: This adversary assigns loss n to one of the experts which received the highest probability by the algorithm and all other experts are assigned loss zero. Clearly the optimal allocation against such an adversary uses the uniform distribution over those experts with zero loss. The number of experts with loss zero is reduced by one in each trial. At trial $t=1,\ldots,n-1$, $n+1-t$ experts are left and the expected loss is $\frac{n}{n+1-t}$. In the first $n-1$ trials the algorithm incurs expected loss

$$\sum_{i=2}^{n}\frac{n}{i}\approx n\ln n.$$

When the loss of the best expert is large then the following theorem follows from Corollary 11:

Theorem 6 *There exists n_0 such that for each dimension $n \geq n_0$, there is a T_n where for any number of trials $T \geq T_n$ the following holds for any algorithm A for learning permutations of n elements in the allocation setting: there is a sequence S of T trials such that*

$$\mathcal{L}_{\text{best}}(S) \leq nT/2 \quad \textit{and} \quad \mathcal{L}_A(S) - \mathcal{L}_{\text{best}}(S) \geq \sqrt{(nT/2)\, n \ln n}.$$

These two lower bounds can be combined to the following lower bound on the expected regret for our permutation learning problem:

$$\max\left(\sqrt{\mathcal{L}_{\text{best}} n \ln n}, n \ln n\right) \geq \frac{\sqrt{\mathcal{L}_{\text{best}} n \ln n} + n \ln n}{2}.$$

This means that the tuned upper bound on the expected regret of PermELearn given after Theorem 4 cannot be improved by more than a small ($2\sqrt{2}$) constant factor.

8. Conclusions

We considered the problem of learning a permutation on-line, when the per-trial loss is specified by a matrix $L \in [0,1]^{n \times n}$ and the loss of a permutation matrix Π is the linear loss $\Pi \bullet L$. The standard approach would treat each permutation as an expert. However this is computationally inefficient and introduces an additional factor of n in the regret bounds (since the per-trial loss of a permutation is $[0,n]$ rather than $[0,1]$). We do not know if this factor of n is necessary for permutations, and it remains open whether their special structure allows better regret bounds on the standard expert algorithms when the experts are permutations.

We developed a new algorithm called PermELearn that uses a doubly stochastic matrix to maintain its uncertainty over the hidden permutation. PermELearn decomposes this doubly stochastic matrix into a small mixture of permutation matrices and predicts with a random permutation from this mixture. A similar decomposition was used by Warmuth and Kuzmin (2008) to learn as well as the best fixed-size subset of experts.

PermELearn belongs to the Exponentiated Gradient family of updates and the analysis uses a relative entropy as a measure of progress. The main technical insight is that the per-trial progress bound already holds for the un-normalized update and that re-balancing the matrix only increases the progress. Since the re-balancing step does not have a closed form, accounting for it in the analysis would otherwise be problematic. We also showed that the update for the Hedge algorithm can be split into an un-normalized update and a normalization. In this more basic setting the per trial progress bound also holds for the un-normalized update.

Our analysis techniques rely on Bregman projection methods[9] and the regret bounds hold not only for permutations but also for mixtures of permutations. This means that if we have additional convex constraints that are satisfied by the mixture that we compare against, then we can project the algorithm's weight matrix onto these constraints without hurting the analysis (Herbster and Warmuth, 2001). With these kinds of side constraints we can enforce some relationships between the parameters, such as $W_{i,j} \geq W_{i,k}$ (i is more likely mapped to j than k).

Our main contribution is showing how to apply the analysis techniques from the expert advice setting to the problem of efficiently learning a permutation. This means that many of the tools from

9. Following Kuzmin and Warmuth (2007), we also showed in Appendix B that the regret bounds proven in this paper can be reproduced with potential based methods.

the expert setting are likely to carry over to permutations: lower bounding the weights when the comparator is shifting (Herbster and Warmuth, 1998), long-term memory when shifting between a small set of comparators (Bousquet and Warmuth, 2002), capping the weights from the top if the goal is to be close to the best set of disjoint permutations of fixed size (Warmuth and Kuzmin, 2008), adapting the updates to the multi-armed bandit setting when less feedback is provided (Auer et al., 2002),[10] and PAC Bayes analysis of the exponential updates (McAllester, 2003).

We also applied the "Follow the Perturbed Leader" techniques to our permutation problem. This algorithm adds randomness to the total losses and then predicts with a minimum weighted matching which costs $O(n^3)$ whereas our more complicated algorithm is at least $O(n^4)$ and has precision issues. However the bounds currently provable for the FPL* algorithm of Kalai and Vempala (2005) are much worse than for our PermELearn algorithm. The key open problem is whether we can have the best of both worlds: add randomness to the loss matrix so that the expected minimum weighted matching is the stochastic matrix produced by the PermELearn update (4). This would mean that we could use the faster algorithm together with our tighter analysis. In the simpler weighted majority setting this has been done already (Kuzmin and Warmuth, 2005; Kalai, 2005). However we do not yet know how to simulate the PermELearn update this way.

Our on-line learning problem requires that the learner's prediction to be an actual permutation. This requirement makes sense for the linear loss we focus on in this paper, but may be less appropriate for on-line regression problems. Consider the case where on each trial the algorithm selects a doubly stochastic matrix M while nature simultaneously picks a matrix $X \in [0,1]^{n\times n}$ and a real number y. The prediction is $\hat{y} = M \bullet X$ and the loss on the trial is $(\hat{y}-y)^2$. With this convex quadratic loss, it is generally better for the algorithm to hedge its bets between competing permutations and select its doubly stochastic parameter matrix W as M instead of a random permutation matrix Π chosen s.t. $\mathbb{E}(\Pi) = W$. The Exponentiated Gradient algorithm can be applied to this type of non-linear regression problem (see, e.g., Helmbold et al., 1999) and Sinkhorn Balancing can project the parameter matrix W onto the row and column sum constraints.

We close with an open problem involving higher order loss functions. In this paper we considered linear losses specified by a square matrix L where $L_{i,j}$ gives the loss when entry (i, j) is used in the permutation. Can one prove good regret bounds when the loss depends on how the permutation assigns multiple elements? A pairwise loss could be represented with a four-dimensional matrix L where $L_{i,j,k,l}$ is added to the loss only when the predicted permutation maps *both* i to j *and* k to l. The recently developed Fourier analysis techniques for permutations (Kondor et al., 2007; Huang et al., 2009) may be helpful in generalizing our techniques to this kind of higher order loss.

Acknowledgments

We thank Vishy Vishwanathan for helping us simplify the lower bounds, and David DesJardins for helpful discussions and pointers to the literature on Sinkhorn Balancing.

10. Recently, a less efficient algorithm which explicitly maintains one expert per permutation has been analyzed in the bandit setting by Cesa-Bianchi and Lugosi (2009). However the bounds they obtain have the loss range as an additional factor in the regret bound (a factor of n for permutations).

Appendix A. Size of the Decomposition

Here we show that the iterative matching method of Algorithm 1 requires most n^2-2n+2 permutations to decompose an doubly stochastic matrix. This matches the bound provided by Birkhoff's Theorem. Note that the discussion in Section 4 shows why Algorithm 1 can always find a suitable permutation.

Theorem 7 *Algorithm 1 decomposes any doubly stochastic matrix into a convex combination of at most n^2-2n+2 permutations.*

Proof Let W be a doubly stochastic matrix and let $\Pi_1,\ldots,\Pi_\ell$ and $\alpha_1,\ldots,\alpha_\ell$ be any sequence of permutations and coefficients created by Algorithm 1 on input W. For $0\le j\le\ell$, define $M^j = W-\sum_{i=1}^{j}\alpha_i\Pi_i$. By permuting rows and columns we can assume without loss of generality that Π_ℓ is the identity permutation. Let G^j (for $1\le j\le\ell$) be the (undirected) graph on the n vertices $\{1,\ldots n\}$ where the undirected edge $\{p,q\}$ between nodes $p\neq q$ is present if and only if either $M^j_{p,q}$ or $M^j_{q,p}$ is non-zero. Thus both $G^{\ell-1}$ and G^ℓ are the empty graph and each G^{j+1} has a (not necessarily strict) subset of the edges in G^j. Note the natural correspondences between vertices in the graphs and rows and columns in the matrices.

The proof is based in the following key invariant:

$$\#\text{ of zero entries in } M^j \ge j + (\#\text{ connected components in } G^j) - 1.$$

This holds for the initial M^0. Furthermore, when the connected components of G^j and G^{j+1} are the same, the algorithm insures that M^{j+1} has at least one more zero than M^j. We now analyze the case when new connected components are created.

Let vertex set V be a connected component in G^{j+1} that was split off a larger connected component in G^j. We overload the notation, and use V also for the set of matrix rows and/or columns associated with the vertices in the connected component.

Since V is a connected component of G^{j+1} there are no edges going between V and the rest of the graph, so if M^{j+1} is viewed as a (conserved) flow, there is no flow either into or out of V:

$$\sum_{r\in V}\sum_{c\notin V} M^{j+1}_{r,c} = \sum_{r\notin V}\sum_{c\in V} M^{j+1}_{r,c} = 0.$$

Thus all entries of M^j in the sets $\{M^j_{r,c}>0 : r\in V, c\notin V\}$ and $\{M^j_{r,c}>0 : r\notin V, c\in V\}$ are set to zero in M^{j+1}. Since V was part of a larger connected component in G^j, at least one of these sets must be non-empty. We now show that both these sets of entries are non-empty.

Each row and column of M^j sum to $1-\sum_{i=1}^{j}\alpha_i$. Therefore

$$\left(1-\sum_{i=1}^{j}\alpha_i\right)|V| = \sum_{r\in V}\sum_{c=1}^{n} M^j_{r,c} = \sum_{c\in V}\sum_{r=1}^{n} M^j_{r,c}.$$

By splitting the inner sums we get:

$$\sum_{r\in V}\sum_{c\in V} M^j_{r,c} + \sum_{r\in V}\sum_{c\notin V} M^j_{r,c} = \sum_{c\in V}\sum_{r\in V} M^j_{r,c} + \sum_{c\in V}\sum_{r\notin V} M^j_{r,c}.$$

By canceling the first sums and viewing M^j as a flow in G^j we conclude that the total flow out of V in M^j equals the total flow into V in M^j, that is,

$$\sum_{r\in V}\sum_{c\notin V} M^j_{r,c} = \sum_{c\in V}\sum_{r\notin V} M^j_{r,c}$$

and both sets $\{M^j_{r,c} > 0 : r \in V, c \notin V\}$ and $\{M^j_{r,c} > 0 : r \notin V, c \in V\}$ sum to the same positive total, and thus are non-empty.

This establishes the following fact that we can use in the remainder of the proof: for each new connected component V in G^{j+1}, some entry $M^j_{r,c}$ from a row r in V was set to zero.

Now let k_j (and k_{j+1}) be the number of connected components in graph G^j (and G^{j+1} respectively). Since the edges in G^{j+1} are a subset of the edges in G^j, $k_{j+1} \geq k_j$. We already verified the invariant when $k_j = k_{j+1}$, so we proceed assuming $k_{j+1} > k_j$. In this case at most $k_j - 1$ components of G^j survive when going to G^{j+1}, and at least $k_{j+1} - (k_j - 1)$ new connected components are created. The vertex sets of the new connected components are disjoint, and in the rows corresponding to each new connected component there is at least one non-zero entry in M^j that is zero in M^{j+1}. Therefore, M^{j+1} has at least $k_{j+1} - k_j + 1$ more zeros than M^j, verifying the invariant for the case when $k_{j+1} > k_j$.

Since $G^{\ell-1}$ has n connected components, the invariant shows that the number of zeros in $M^{\ell-1}$ is at least $\ell - 1 + n - 1$. Furthermore, M^ℓ has n more zeros than $M^{\ell-1}$, so M^ℓ has at least $\ell + 2n - 2$ zeros. Since M^ℓ has only n^2 entries, $n^2 \geq \ell + 2n - 2$ and $\ell \leq n^2 - 2n + 2$ as desired. ■

The fact that Algorithm 1 uses at most $n^2 - 2n + 2$ permutations can also be established with a dimensionality argument like that in Section 2.7 of Bazaraa et al. (1977).

Appendix B. Potential Based Bounds

Let us begin with the on-line allocation problem in the simpler expert setting. There are always two ways to motivate on-line updates. One trades the divergence to the last weight vector against the loss in the last trial, and the other trades the divergence to the initial weight vector against the loss in all past trials (Azoury and Warmuth, 2001):

$$w^t := \operatorname*{argmin}_{\sum_i w_i = 1} \left(\Delta(w, w^{t-1}) + \eta\, w \cdot \ell^t\right), \quad w^t := \operatorname*{argmin}_{\sum_i w_i = 1} \left(\Delta(w, w^0) + \eta\, w \cdot \ell^{\leq t}\right).$$

By differentiating the Lagrangian for each optimization problem we obtain the solutions to both minimization problems:

$$w^t_i = w^{t-1}_i e^{-\eta \ell^t_i + \widetilde{\beta}^t}, \quad w^t_i = w^0_i e^{-\eta \ell^{\leq t}_i + \beta^t},$$

where the signs of the Lagrange multipliers $\widetilde{\beta}^t$ and β^t are unconstrained and their values are chosen so that the equality constraints are satisfied. The left update can be unrolled to obtain

$$w^t_i = w^0_i e^{-\eta \ell^{\leq t}_i + \sum_{q=1}^t \widetilde{\beta}^q}.$$

This means the Lagrangian multipliers for both problems are related by the equality $\sum_{q=1}^t \widetilde{\beta}^q = \beta^t$ and both problems have the same solution:[11] $w^t_i = \frac{w^0_i e^{-\eta \ell^{\leq t}_i}}{\sum_{j=1}^n w^0_j e^{-\eta \ell^{\leq t}_j}}$. We use the value of the right convex

11. The solutions can differ if the minimization is over linear inequality constraints (Kuzmin and Warmuth, 2007).

optimization problem's objective function as our potential v^t. Its Lagrangian is

$$\sum_i \left(w_i \ln \frac{w_i}{w_i^0} + w_i^0 - w_i + \eta w_i \ell_i^{\leq t} \right) + \beta \left(\sum_i w_i - 1 \right)$$

and since there is no duality gap:[12]

$$v^t := \min_{\sum_i w_i = 1} \left(\Delta(w, w^0) + \eta \, w \cdot \ell_{\leq t} \right) = \max_{\beta} \underbrace{\sum_i w_i^0 (1 - e^{-\eta \ell_i^{\leq t} - \beta}) - \beta}_{\text{dual function } \theta^t(\beta)}.$$

Here β is the (unconstrained) dual variable for the primal equality constraint and the w_i's have been optimized out. By differentiating we can optimize β in the dual problem and arrive at

$$v^t = -\ln \sum_i w_i^0 e^{-\eta \ell_i^{\leq t}}.$$

This form of the potential has been used extensively for analyzing expert algorithms (see, e.g., Kivinen and Warmuth, 1999; Cesa-Bianchi and Lugosi, 2006). One can easily show the following key inequality (essentially Lemma 5.2 of Littlestone and Warmuth, 1994):

$$\begin{aligned} v^t - v^{t-1} &= -\ln \sum_i w_i^0 e^{-\eta \ell_i^{\leq t}} + \ln \sum_i w_i^0 e^{-\eta \ell_i^{<t}} \\ &= -\ln \sum_i w_i^{t-1} e^{-\eta \ell_i^t} \\ &\geq -\ln \sum_i w_i^{t-1} (1 - (1 - e^{-\eta}) \ell_i^t) \\ &\geq (1 - e^{-\eta}) \, w^{t-1} \cdot \ell^t. \end{aligned} \tag{13}$$

Summing over all trials and using $v^0 = 0$ gives the familiar bound:

$$\sum_{t=1}^{T} w^{t-1} \cdot \ell^t \leq \frac{v^T}{1 - e^{-\eta}} = \frac{1}{1 - e^{-\eta}} \min_{\sum_i w_i = 1} \left(\Delta(w, w^0) + \eta \, w \cdot \ell^{\leq T} \right).$$

Note that by Kivinen and Warmuth (1999)

$$v^t - v^{t-1} = -\ln \left(\sum_i w_i^{t-1} e^{-\eta \ell_i^t} \right) = \Delta(u, w^{t-1}) - \Delta(u, w^t) + \eta \, u \cdot \ell^t,$$

and therefore the Inequality (13) is the same as Inequality (10). Since

$$\sum_{t=1}^{T} (v^t - v^{t-1}) = v^T = -\ln \left(\sum_i w_i^0 e^{-\eta \ell_i^{\leq t}} \right),$$

summing the bound (13) over t coincides with the bound (11).

12. There is no duality gap in this case because the primal problem is a feasible convex optimization problem subject to linear constraints.

We now reprove the key inequality (13) using the dual function $\theta^t(\beta)$. Note β^t maximizes this function, that is, $v^t = \theta^t(\beta^t)$, and the optimal primal solution is $w_i^t = w_i^0 e^{-\eta \ell_i^{\leq t} - \beta^t}$. Now,

$$\begin{aligned} v^t - v^{t-1} &= \theta^t(\beta^t) - \theta^{t-1}(\beta^{t-1}) \\ &\geq \theta^t(\beta^{t-1}) - \theta^{t-1}(\beta^{t-1}) \\ &= \sum_i \underbrace{w_i^0 e^{-\eta \ell_i^{<t} - \beta^{t-1}}}_{w_i^{t-1}} (1 - e^{-\eta \ell_i^t}) \\ &\geq \sum_i w_i^{t-1} (1 - (1 - (1 - e^{-\eta})\ell_i^t)) \\ &= (1 - e^{-\eta})\, w^{t-1} \cdot \ell_t \end{aligned}$$

where we used $e^{-\eta \ell_i^t} \leq 1 - (1 - e^{-\eta})\ell_i^t$ to get the fourth line. Notice that in the first inequality above we used $\theta^t(\beta^t) \geq \theta^t(\beta^{t-1})$. This is true because β^t maximizes $\theta^t(\beta)$ and the old choice β^{t-1} is non-optimal. The dual parameter β^{t-1} assures that w^{t-1} is normalized and $\theta^t(\beta^{t-1})$ is related to plugging the intermediate unnormalized weights $w'^t_i := w_i^0 e^{-\eta \ell_i^{\leq t} - \beta^{t-1}}$ into the primal problem for trial t. This means that the inequality $\theta^t(\beta^t) \geq \theta^t(\beta^{t-1})$ corresponds to the Bregman projection of the unnormalized update onto the equality constraint. The difference $\theta^t(\beta^{t-1}) - \theta^{t-1}(\beta^{t-1})$ in the second line above is the progress in the value when going from w^{t-1} at the end of trial $t-1$ to the intermediate unnormalized update w'^t at trial t. Therefore this proof also does not exploit the normalization.

The bound for the permutation problem follows the same outline. We use the value of the following optimization problem as our potential:

$$\begin{aligned} v_{t+1} &:= \min_{\substack{\forall i : \sum_j A_{i,j} = 1 \\ \forall j : \sum_i A_{i,j} = 1}} \left(\Delta(A, W^0) + \eta\, (A \bullet L^{\leq t})\right) \\ &= \max_{\alpha_i, \beta_j} \underbrace{\sum_{i,j} W_{i,j}^0 (1 - e^{-\eta L_{i,j}^{\leq t} - \alpha_i - \beta_j}) - \sum_i \alpha_i - \sum_j \beta_j}_{\theta^t(\alpha, \beta)}. \end{aligned}$$

The α_i and β_j are the dual variables for the row and column constraints. Now we can't optimize out the dual variables in the dual function $\theta^t(\alpha, \beta)$ does not have a maximum in closed form. Nevertheless the above proof technique based on duality still works. Let α^t and β^t be the optimizers of $\theta^t(\alpha, \beta)$. Then the optimum primal solution (the parameter vector of PermELearn) becomes

$$W_{i,j}^t = W_{i,j}^0 e^{-\eta L_{i,j}^{\leq t} - \alpha_i^t - \beta_j^t}$$

and we can analyze the increase in value as before:

$$\begin{aligned} v^t - v^{t-1} &= \theta^t(\alpha^t, \beta^t) - \theta^{t-1}(\alpha^{t-1}, \beta^{t-1}) \\ &\geq \theta^t(\alpha_{t-1}, \beta_{t-1}) - \theta^{t-1}(\alpha^{t-1}, \beta^{t-1}) \\ &= \sum_{i,j} \underbrace{W_{i,j}^0 e^{-\eta L_{i,j}^{<t} - \alpha_i^{t-1} - \beta_j^{t-1}}}_{W_{i,j}^{t-1}} (1 - e^{-\eta L_{i,j}^{t-1}}) \\ &\geq \sum_{i,j} W_{i,j}^{t-1} (1 - (1 - e^{-\eta}) L_{i,j}^t) \\ &= (1 - e^{-\eta})\, W^{t-1} \bullet L^t. \end{aligned}$$

Summing over trials in the usual way gives the bound

$$\sum_{t=1}^{T} W^{t-1} \bullet L^t \leq \frac{v_T}{1-e^{-\eta}} = \frac{1}{1-e^{-\eta}} \min_{\substack{\forall i: \sum_j A_{i,j}=1 \\ \forall j: \sum_i A_{i,j}=1}} \left(\Delta(A, W^0) + \eta\, (A \bullet L_{\leq T})\right)$$

which is the same as the bound of Theorem 4.

Appendix C. Lower Bounds for the Expert Advice Setting

We first modify a known lower bound from the expert advice setting with the absolute loss (Cesa-Bianchi et al., 1997). We begin by describing that setting and show how it relates to the allocation setting for experts.

In the expert advice setting there are n experts. Each trial t starts with nature selecting a *prediction* x_i^t in [0,1] for each expert $i \in \{1,\ldots,n\}$. The algorithm is given these predictions and then produces its own prediction $\hat{y}^t \in [0,1]$. Finally, nature selects a *label* $y^t \in \{0,1\}$ for the trial. The algorithm is charged loss $|\hat{y}^t - y^t|$ and expert i gets loss $|x_i^t - y^t|$.

Any algorithm in the allocation setting leads to an algorithm in the above expert advice setting: keep the weight update unchanged, predict with the weighted average (i.e., $\hat{y}^t = w^{t-1} \cdot \mathbf{x}^t$) and define the loss vector in $\ell^t \in [0,1]^n$ in terms of the absolute loss:

$$|\underbrace{w^{t-1} \cdot \mathbf{x}^t}_{\hat{y}^t} - y^t| \;=\; \sum_i w_i^{t-1} \underbrace{|x_i^t - y^t|}_{\ell_i^t} \;=\; w^{t-1} \cdot \ell^t,$$

where the first equality holds because $x_i^t \in [0,1]$ and $y^t \in \{0,1\}$. This means that any lower bound on the regret in the above expert advice setting immediately leads to a lower bound on the expected loss in the allocation setting for experts when the loss vectors lie in $[0,1]^n$.

We now introduce some more notation and state the lower bound from the expert advice setting that we build on. Let $\mathcal{S}_{n,T}$ be the set of all sequences of T trials with n experts in the expert advice setting with the absolute loss. Let $V_{n,T}$ be the minimum over algorithms of the worst case regret over sequences in $\mathcal{S}_{n,T}$.

Theorem 8 *(Cesa-Bianchi et al., 1997, Theorem 4.5.2)*

$$\lim_{n\to\infty} \lim_{T\to\infty} \frac{V_{n,T}}{\sqrt{(T/2)\ln n}} = 1.$$

This means that for all $\varepsilon > 0$ there exists n_ε such that for each $n \geq n_\varepsilon$, there is a $T_{\varepsilon,n}$ where for all $T \geq T_{\varepsilon,n}$,

$$V_{n,T} \geq (1-\varepsilon)\sqrt{(T/2)\ln n}.$$

By further expanding the definition of $V_{n,T}$ we get the following version of the above lower bound that avoids the use of limits:

Corollary 9 *For all $\varepsilon > 0$ there exists n_ε such that for each number of experts $n \geq n_\varepsilon$, there is a $T_{\varepsilon,n}$ where for any number of trials $T \geq T_{\varepsilon,n}$ the following holds for any algorithm A in the expert advice setting with the absolute loss: there is a sequence S of T trials with n experts such that*

$$\mathcal{L}_A(S) - \mathcal{L}_{\text{best}}(S) \geq (1-\varepsilon)\sqrt{(T/2)\ln n}.$$

This lower bound on the regret depends on the number of trials T. We now use a reduction to bound $\mathcal{L}_{\text{best}}(S)$ by $T/2$. Define $R(S^-)$ as the transformation that takes a sequence S^- of trials in $\mathcal{S}_{n-1,T}$ and produces a sequence of trials in $\mathcal{S}_{n,T}$ by adding an extra expert whose predictions are simply 1 minus the predictions of the first expert. On each trial the absolute loss of the additional expert on sequence $R(S^-)$ is 1 minus the loss of the first expert. Therefore either the first expert or the additional expert will have loss at most $T/2$ on $R(S^-)$.

Theorem 10 *For all $\varepsilon > 0$ there exists n_ε such that for each number of experts $n \geq n_\varepsilon$, there is a $T_{\varepsilon,n}$ where for any number of trials $T \geq T_{\varepsilon,n}$ the following holds for any algorithm A in the expert advice setting with the absolute loss: there is a sequence S of T trials with n experts such that*

$$\mathcal{L}_{\text{best}}(S) \leq T/2 \quad \textit{and} \quad \mathcal{L}_A(S) - \mathcal{L}_{\text{best}}(S) \geq (1-\varepsilon)\sqrt{(T/2)\ln n}.$$

Proof We begin by showing that the regret on a transformed sequence in $\{R(S^-) : S^- \in \mathcal{S}_{n-1,T}\}$ is at least $(1-\varepsilon/2)\sqrt{(T/2)\ln(n-1)}$.

Note that for all $R(S^-)$, $\mathcal{L}_{\text{best}}(R(S^-)) \leq T/2$ and assume to the contrary that some algorithm A has regret strictly less than $(1-\varepsilon/2)\sqrt{(T/2)\ln(n-1)}$ on every sequence in $\{R(S^-) : S^- \in \mathcal{S}_{n-1,T}\}$. We then create an algorithm A^- that runs transformation $R(\cdot)$ on-the-fly and predicts as A does on the transformed sequence. Therefore A^- on S^- and A on $R(S^-)$ make the same predictions and have the same total loss. On every sequence $S^- \in \mathcal{S}_{n-1,T}$ we have $\mathcal{L}_{\text{best}}(S^-) \geq \mathcal{L}_{\text{best}}(R(S))$ and therefore

$$\begin{aligned}
\mathcal{L}_{A^-}(S^-) - \mathcal{L}_{\text{best}}(S^-) &\leq \mathcal{L}_{A^-}(S^-) - \mathcal{L}_{\text{best}}(R(S^-)) \\
&= \mathcal{L}_A(R(S^-)) - \mathcal{L}_{\text{best}}(R(S^-)) \\
&< (1-\varepsilon/2)\sqrt{(T/2)\ln(n-1)}.
\end{aligned}$$

Now if $n-1$ is at least the $n_{\varepsilon/2}$ of Corollary 9 and T is at least the $T_{\varepsilon/2,n-1}$ of the same corollary, then this contradicts that corollary.

This means that for any algorithm A and large enough n and T, there is a sequence S for which the algorithm has regret at least $(1-\varepsilon/2)\sqrt{(T/2)\ln(n-1)}$ and $\mathcal{L}_{\text{best}}(S) \leq T/2$. By choosing the lower bound on n large enough,

$$(1-\varepsilon/2)\sqrt{(T/2)\ln(n-1)} \geq (1-\varepsilon)\sqrt{(T/2)\ln n}$$

and the theorem follows. ■

Note that the tuned upper bounds in the allocation setting (9) have an additional factor of $\sqrt{2}$. This is due to the fact that in the allocation setting the algorithm predicts with the weighted average and this is non-optimal. In the expert setting with the absolute loss, the upper bound (based on a different prediction function) and the lower bound on the regret are asymptotically tight (See Theorem 8). We are now ready to prove our lower bound for the allocation setting with experts when the losses of the experts are in $[0,n]^n$ instead of $[0,1]^n$.

Corollary 11 *There exists n_0 such that for each dimension $n \geq n_0$, there is a T_n where for any number of trials $T \geq T_n$ the following holds for any algorithm A for allocation setting with n experts: there is a sequence S of T trials with loss vectors in $[0,n]^n$ such that*

$$\mathcal{L}_{\text{best}}(S) \leq nT/2 \quad \textit{and} \quad \mathcal{L}_A(S) - \mathcal{L}_{\text{best}}(S) \geq \sqrt{(nT/2)n\ln n}.$$

Proof Via the reduction we stated at the beginning of the section, the following lower bound for the allocation setting with n experts immediately follows from the previous theorem: For any algorithm in the allocation setting for n experts there is a sequence $\widetilde{S}$ of T trials where the losses of the experts lie in $[0,1]$ such that

$$\mathcal{L}_{\text{best}}(\widetilde{S}) \leq T/2 \quad \text{and} \quad \mathcal{L}_A(\widetilde{S}) - \mathcal{L}_{\text{best}}(\widetilde{S}) \geq \sqrt{(T/2)\ln n}.$$

Now we simply scale the loss vectors by the factor n, that is, the scaled sequences S have loss vectors in the range $[0,n]^n$ and $\mathcal{L}_{\text{best}}(S) \leq nT/2$. The lower bound becomes $n\sqrt{(T/2)\ln n} = \sqrt{(nT/2)n\ln n}$. ■

References

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

K. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Journal of Machine Learning*, 43(3):211–246, June 2001. Special issue on *Theoretical Advances in On-line Learning, Game Theory and Boosting*, edited by Yoram Singer.

H. Balakrishnan, I. Hwang, and C. Tomlin. Polynomial approximation algorithms for belief matrix maintenance in identity management. In *43rd IEEE Conference on Decision and Control*, pages 4874–4879, December 2004.

M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows*. Wiley, second edition, 1977.

R. Bhatia. *Matrix Analysis*. Springer-Verlag, Berlin, 1997.

A. Blum, S. Chawla, and A. Kalai. Static optimality and dynamic search-optimality in lists and trees. *Algorithmica*, 36:249–260, 2003.

O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3:363–396, 2002.

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Physics*, 7:200–217, 1967.

Y. Censor and A. Lent. An iterative row-action method for interval convex programming. *Journal of Optimization Theory and Applications*, 34(3):321–353, July 1981.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT 09)*, 2009.

N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, May 1997.

Y. Chen, L. Fortnow, N. Lambert, D. Pennock, and J. Wortman. Complexity of combinatorial market makers. In *Ninth ACM Conference on Electronic Commerce (EC '08)*. ACM Press, July 2008.

J. Franklin and J. Lorenz. On the scaling of multidimensional matrices. *Linear Algebra and its applications*, 114/115:717–735, 1989.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

M. Fürer. Quadratic convergence for scaling of matrices. In *Proceedings of ALENEX/ANALCO*, pages 216–223. SIAM, 2004.

Geoffrey J. Gordon. No-regret algorithms for online convex programs. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *NIPS*, pages 489–496. MIT Press, 2006.

D. P. Helmbold and R. E. Schapire. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27(01):51–68, 1997.

D. P. Helmbold, J. Kivinen, and M. K. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291–1304, November 1999.

M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998. Earlier version in 12th ICML, 1995.

M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

J. Huang, C. Guestrin, and L. Guibas. Fourier theoretic probabilistic inference over permutations. *Journal of Machine Learning Research*, 10:997–1070, 2009.

M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4):671–697, July 2004.

A. Kalai. Simulating weighted majority with FPL. Private communication, 2005.

A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, 2005. Special issue Learning Theory 2003.

B. Kalantari and L. Khachiyan. On the complexity of nonnegative-matrix scaling. *Linear Algebra and its applications*, 240:87–103, 1996.

J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, 132(1):1–64, January 1997.

J. Kivinen and M. K. Warmuth. Averaging expert predictions. In *Computational Learning Theory, 4th European Conference (EuroCOLT '99), Nordkirchen, Germany, March 29-31, 1999, Proceedings*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 153–167. Springer, 1999.

R. Kondor, A. Howard, and T. Jebara. Multi-object tracking with representations of the symmetric group. In *Proc. of the 11th International Conference on Artificial Intelligence and Statistics*, March 2007.

D. Kuzmin and M. K. Warmuth. Optimum follow the leader algorithm. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT '05)*, pages 684–686. Springer-Verlag, June 2005. Open problem.

D. Kuzmin and M. K. Warmuth. Online Kernel PCA with entropic matrix updates. In *Proceedings of the 24rd international conference on Machine learning (ICML '07)*, pages 465–471. ACM International Conference Proceedings Series, June 2007.

N. Linial, A. Samorodnitsky, and A. Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Combinatorica*, 20(4):545–568, 2000.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Inform. Comput.*, 108(2): 212–261, 1994. Preliminary version in FOCS '89.

D. McAllester. PAC-Bayesian stochastic model selection. *Machine Learning*, 51(1):5–21, 2003.

R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Staticstics*, 35(2):876–879, June 1964.

E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4:773–818, 2003.

V. Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.

M. K. Warmuth and D. Kuzmin. Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9:2217–2250, 2008.

M. K. Warmuth and S.V.N. Vishwanathan. Leaving the span. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT '05)*, Bertinoro, Italy, June 2005. Springer-Verlag. Journal version in progress.

Journal of Machine Learning Research 10 (2009) 1737-1754 Submitted 1/09; Revised 4/09; Published 7/09

SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent

Antoine Bordes[*] ANTOINE.BORDES@LIP6.FR
LIP6, Université Pierre et Marie Curie
104, Avenue du Président Kennedy
75016 Paris, France

Léon Bottou LEONB@NEC-LABS.COM
NEC Laboratories America, Inc.
4 Independence Way
Princeton, NJ 08540, USA

Patrick Gallinari PATRICK.GALLINARI@LIP6.FR
LIP6, Université Pierre et Marie Curie
104, Avenue du Président Kennedy
75016 Paris, France

Editors: Soeren Sonnenburg, Vojtech Franc, Elad Yom-Tov and Michele Sebag

Abstract

The SGD-QN algorithm is a stochastic gradient descent algorithm that makes careful use of second-order information and splits the parameter update into independently scheduled components. Thanks to this design, SGD-QN iterates nearly as fast as a first-order stochastic gradient descent but requires less iterations to achieve the same accuracy. This algorithm won the "Wild Track" of the first PASCAL Large Scale Learning Challenge (Sonnenburg et al., 2008).

Keywords: support vector machine, stochastic gradient descent

1. Introduction

The last decades have seen a massive increase of data quantities. In various domains such as biology, networking, or information retrieval, fast classification methods able to scale on millions of training instances are needed. Real-world applications demand learning algorithms with low time and memory requirements. The first PASCAL Large Scale Learning Challenge (Sonnenburg et al., 2008) was designed to identify which machine learning techniques best address these new concerns. A generic evaluation framework and various data sets have been provided. Evaluations were carried out on the basis of various performance curves such as training time versus test error, data set size versus test error, and data set size versus training time.[1]

Our entry in this competition, named SGD-QN, is a carefully designed *Stochastic Gradient Descent* (SGD) for *linear Support Vector Machines* (SVM).

Nonlinear models could in fact reach much better generalization performance on most of the proposed data sets. Unfortunately, even in the Wild Track case, the evaluation criteria for the competition reward good scaling properties and short training durations more than they punish suboptimal test errors. Nearly all the competitors chose to implement linear models in order to avoid the

∗. Also at NEC Laboratories America, Inc.

1. This material and its documentation can be found at `http://largescale.first.fraunhofer.de/`.

additional penalty implied by nonlinearities. Although SGD-QN can work on nonlinear models,[2] we only report its performance in the context of linear SVMs.

Stochastic algorithms are known for their poor optimization performance. However, in the large scale setup, when the bottleneck is the computing time rather than the number of training examples, Bottou and Bousquet (2008) have shown that stochastic algorithms often yield the best generalization performances in spite of being worst optimizers. SGD algorithms were therefore a natural choice for the "Wild Track" of the competition which focuses on the relation between training time and test performance.

SGD algorithms have been the object of a number of recent works. Bottou (2007) and Shalev-Shwartz et al. (2007) demonstrate that the plain Stochastic Gradient Descent yields particularly effective algorithms when the input patterns are very sparse, taking less than $O(d)$ space and time per iteration to optimize a system with d parameters. It can greatly outperform sophisticated batch methods on large data sets but suffers from slow convergence rates especially on ill-conditioned problems. Various remedies have been proposed: *Stochastic Meta-Descent* (Schraudolph, 1999) heuristically determines a learning rate for each coefficient of the parameter vector. Although it can solve some ill-conditioning issues, it does not help much for linear SVMs. *Natural Gradient Descent* (Amari et al., 2000) replaces the learning rate by the inverse of the Riemannian metric tensor. This quasi-Newton stochastic method is statistically efficient but is penalized in practice by the cost of storing and manipulating the metric tensor. *Online BFGS* (oBFGS) and *Online Limited storage BFGS* (oLBFGS) (Schraudolph et al., 2007) are stochastic adaptations of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization algorithm. The limited storage version of this algorithm is a quasi-Newton stochastic method whose cost by iteration is a small multiple of the cost of a standard SGD iteration. Unfortunately this penalty is often bigger than the gains associated with the quasi-Newton update. *Online Dual Solvers* for SVMs (Bordes et al., 2005; Hsieh et al., 2008) have also shown good performance on large scale data sets. These solvers can be applied to both linear and nonlinear SVMs. In the linear case, these dual algorithms are surprising close to SGD but do not require fiddling with learning rates. Although this is often viewed as an advantage, we feel that this aspect restricts the improvement opportunities.

The contributions of this paper are twofold:

1. We conduct an analysis of different factors, ranging from algorithmic refinements to implementation details, which can affect the learning speed of SGD algorithms.

2. We present a novel algorithm, denoted SGD-QN, that carefully exploits these speedup opportunities. We empirically validate its properties by benchmarking it against state-of-the-art SGD solvers and by summarizing its results at the PASCAL Large Scale Learning Challenge (Sonnenburg et al., 2008).

The paper is organized as follows: Section 2 analyses the potential gains of quasi-Newton techniques for SGD algorithms. Sections 3 and 4 discuss the sparsity and implementation issues. Finally Section 5 presents the SGD-QN algorithm, and Section 6 reports experimental results.

2. Stochastic gradient works well in models with nonlinear parametrization. For SVMs with nonlinear kernels, we would prefer dual methods, (e.g., Bordes et al., 2005), which can exploit the sparsity of the kernel expansion.

2. Analysis

This section describes our notations and summarizes theoretical results that are relevant to the design of a fast variant of stochastic gradient algorithms.

2.1 SGD for Linear SVMs

Consider a binary classification problem with examples $z = (\mathbf{x}, y) \in \mathbb{R}^d \times \{-1, +1\}$. The linear SVM classifier is obtained by minimizing the primal cost function

$$\mathcal{P}_n(\mathbf{w}) \;=\; \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(y_i \mathbf{w}^\top \mathbf{x}_i) \;=\; \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\lambda}{2}\|\mathbf{w}\|^2 + \ell(y_i \mathbf{w}^\top \mathbf{x}_i)\right), \tag{1}$$

where the hyper-parameter $\lambda > 0$ controls the strength of the regularization term. Although typical SVMs use mildly non regular convex loss functions, we assume in this paper that the loss $\ell(s)$ is convex and twice differentiable with continuous derivatives ($\ell \in C^2[\mathbb{R}]$). This could be simply achieved by smoothing the traditional loss functions in the vicinity of their non regular points.

Each iteration of the SGD algorithm consists of drawing a random training example $(\mathbf{x}_t, y_t)$ and computing a new value of the parameter $\mathbf{w}_t$ as

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t - \frac{1}{t+t_0}\mathbf{B}\,\mathbf{g}_t(\mathbf{w}_t) \quad \text{where} \quad \mathbf{g}_t(\mathbf{w}_t) = \lambda \mathbf{w}_t + \ell'(y_t \mathbf{w}_t^\top \mathbf{x}_t)\, y_t\, \mathbf{x}_t \tag{2}$$

where the *rescaling matrix* $\mathbf{B}$ is positive definite. Since the SVM theory provides simple bounds on the norm of the optimal parameter vector (Shalev-Shwartz et al., 2007), the positive constant t_0 is heuristically chosen to ensure that the first few updates do not produce a parameter with an implausibly large norm.

- The traditional first-order SGD algorithm, with decreasing learning rate, is obtained by setting $\mathbf{B} = \lambda^{-1}\mathbf{I}$ in the generic update (2):

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t - \frac{1}{\lambda(t+t_0)}\,\mathbf{g}_t(\mathbf{w}_t)\,. \tag{3}$$

- The second-order SGD algorithm is obtained by setting $\mathbf{B}$ to the inverse of the Hessian Matrix $\mathbf{H} = [\,\mathcal{P}_n''(\mathbf{w}_n^*)\,]$ computed at the optimum $\mathbf{w}_n^*$ of the primal cost $\mathcal{P}_n(\mathbf{w})$:

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t - \frac{1}{t+t_0}\mathbf{H}^{-1}\,\mathbf{g}_t(\mathbf{w}_t)\,. \tag{4}$$

Randomly picking examples could lead to expensive random accesses to the slow memory. In practice, one simply performs sequential passes over the randomly shuffled training set.

2.2 What Matters Are the Constant Factors

Bottou and Bousquet (2008) characterize the asymptotic learning properties of stochastic gradient algorithms in the *large scale regime*, that is, when the bottleneck is the computing time rather than the number of training examples.

The first three columns of Table 2.2 report the time for a single iteration, the number of iterations needed to reach a predefined accuracy ρ, and their product, the time needed to reach accuracy ρ.

Stochastic Gradient Algorithm	Cost of one iteration	Iterations to reach ρ	Time to reach accuracy ρ	Time to reach $\mathcal{E} \leq c\,(\mathcal{E}_{\text{app}} + \varepsilon)$
1^{st} Order SGD	$O(d)$	$\frac{\nu\kappa^2}{\rho} + o\left(\frac{1}{\rho}\right)$	$O\left(\frac{d\nu\kappa^2}{\rho}\right)$	$O\left(\frac{d\nu\kappa^2}{\varepsilon}\right)$
2^{nd} Order SGD	$O(d^2)$	$\frac{\nu}{\rho} + o\left(\frac{1}{\rho}\right)$	$O\left(\frac{d^2\nu}{\rho}\right)$	$O\left(\frac{d^2\nu}{\varepsilon}\right)$

Table 1: Asymptotic results for stochastic gradient algorithms, reproduced from Bottou and Bousquet (2008). Compare the second last column (time to optimize) with the last column (time to reach the excess test error ε). *Legend*: n number of examples; d parameter dimension; c positive constant that appears in the generalization bounds; κ condition number of the Hessian matrix $\mathbf{H}$; $\nu = \mathbf{tr}\left(\mathbf{G}\mathbf{H}^{-1}\right)$ with $\mathbf{G}$ the Fisher matrix (see Theorem 1 for more details). The implicit proportionality coefficients in notations $O()$ and $o()$ are of course independent of these quantities.

The excess test error $\mathcal{E}$ measures how much the test error is worse than the best possible error for this problem. Bottou and Bousquet (2008) decompose the test error as the sum of three terms $\mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}}$. The *approximation error* $\mathcal{E}_{\text{app}}$ measures how closely the chosen family of functions can approximate the optimal solution, the *estimation error* $\mathcal{E}_{\text{est}}$ measures the effect of minimizing the empirical risk instead of the expected risk, the *optimization error* $\mathcal{E}_{\text{opt}}$ measures the impact of the approximate optimization on the generalization performance.

The fourth column of Table 2.2 gives the time necessary to reduce the excess test error $\mathcal{E}$ below a target that depends on $\varepsilon > 0$. This is the important metric because the test error is the measure that matters in machine learning.

Both the first-order and the second-order SGD require a time inversely proportional to ε to reach the target test error. Only the constants differ. The second-order algorithm is insensitive to the condition number κ of the Hessian matrix but suffers from a penalty proportional to the dimension d of the parameter vector. Therefore, algorithmic changes that exploit the second-order information in SGD algorithms are unlikely to yield superlinear speedups. We can at best improve the constant factors.

This property is not limited to SGD algorithms. To reach an excess error ε, the most favorable generalization bounds suggest that one needs a number of examples proportional to $1/\varepsilon$. Therefore, the time complexity of any algorithm that processes a non vanishing fraction of these examples cannot scale better than $1/\varepsilon$. In fact, Bottou and Bousquet (2008) obtain slightly worse scaling laws for typical non-stochastic gradient algorithms.

2.3 Limited Storage Approximations of Second-Order SGD

Since the second-order SGD algorithm is penalized by the high cost of performing the update (2) using a full rescaling matrix $\mathbf{B} = \mathbf{H}^{-1}$, it is tempting to consider matrices that admit a sparse representation and yet approximate the inverse Hessian well enough to reduce the negative impact of the condition number κ.

The following theorem describes how the convergence speed of the generic SGD algorithm (2) is related to the spectrum of matrix $\mathbf{HB}$.

Theorem 1 *Let $\mathbb{E}_\sigma$ denote the expectation with respect to the random selection of the examples $(\mathbf{x}_t, y_t)$ drawn independently from the training set at each iteration. Let $\mathbf{w}_n^* = \arg\min_{\mathbf{w}} \mathcal{P}_n(\mathbf{w})$ be an optimum of the primal cost. Define the Hessian matrix $\mathbf{H} = \partial^2 \mathcal{P}_n(\mathbf{w}_n^*)/\partial \mathbf{w}^2$ and the Fisher matrix $\mathbf{G} = \mathbf{G}_t = \mathbb{E}_\sigma\left[\mathbf{g}_t'(\mathbf{w}_n^*)\,\mathbf{g}_t'(\mathbf{w}_n^*)^\top\right]$. If the eigenvalues of $\mathbf{HB}$ are in range $\lambda_{\max} \geq \lambda_{\min} > 1/2$, and if the SGD algorithm* (2) *converges to $\mathbf{w}_n^*$, the following inequality holds:*

$$\frac{\mathbf{tr}\,(\mathbf{HBGB})}{2\lambda_{\max}-1}\,t^{-1} + \mathrm{o}\left(t^{-1}\right) \leq \mathbb{E}_\sigma\left[\mathcal{P}_n(\mathbf{w}_t) - \mathcal{P}_n(\mathbf{w}_n^*)\right] \leq \frac{\mathbf{tr}\,(\mathbf{HBGB})}{2\lambda_{\min}-1}\,t^{-1} + \mathrm{o}\left(t^{-1}\right).$$

The proof of the theorem is provided in the appendix. Note that the theorem assumes that the generic SGD algorithm converges. Convergence in the first-order case holds under very mild assumptions (e.g., Bottou, 1998). Convergence in the generic SGD case holds because it reduces to the first-order case with a the change of variable $\mathbf{w} \to \mathbf{B}^{-\frac{1}{2}}\,\mathbf{w}$. Convergence also holds under slightly stronger assumptions when the rescaling matrix $\mathbf{B}$ changes over time (e.g., Driancourt, 1994).

The following two corollaries recover the maximal number of iterations listed in Table 2.2 with $\nu = \mathbf{tr}\left(\mathbf{GH}^{-1}\right)$ and $\kappa = \lambda^{-1}\|\mathbf{H}\|$. Corollary 2 gives a very precise equality for the second-order case because the lower bound and the upper bound of the theorem take identical values. Corollary 3 gives a much less refined bound in the first-order case.

Corollary 2 *Assume $\mathbf{B} = \mathbf{H}^{-1}$ as in the second-order SGD algorithm* (4)*. Under the assumptions of Theorem 1, we have*

$$\mathbb{E}_\sigma\left[\mathcal{P}_n(\mathbf{w}_t) - \mathcal{P}_n(\mathbf{w}_n^*)\right] \;=\; \mathbf{tr}\left(\mathbf{GH}^{-1}\right)\,t^{-1} + \mathrm{o}\left(t^{-1}\right) \;=\; \nu t^{-1} + \mathrm{o}\left(t^{-1}\right).$$

Corollary 3 *Assume $\mathbf{B} = \lambda^{-1}\,\mathbf{I}$ as in the first-order SGD algorithm* (3)*. Under the assumptions of Theorem 1, we have*

$$\mathbb{E}_\sigma\left[\mathcal{P}_n(\mathbf{w}_t) - \mathcal{P}_n(\mathbf{w}_n^*)\right] \;\leq\; \lambda^{-2}\,\mathbf{tr}\left(\mathbf{H}^2\mathbf{GH}^{-1}\right)\,t^{-1} + \mathrm{o}\left(t^{-1}\right) \;\leq\; \kappa^2 \nu t^{-1} + \mathrm{o}\left(t^{-1}\right).$$

An often rediscovered property of second order SGD provides an useful point of reference:

Theorem 4 (Fabian, 1973; Murata and Amari, 1999; Bottou and LeCun, 2005)
Let $\mathbf{w}^* = \arg\min \frac{\lambda}{2}\|\mathbf{w}\|^2 + \mathbb{E}_{\mathbf{x},y}\left[\,\ell(y\,\mathbf{w}^\top\mathbf{x})\,\right]$. Given a sample of n independent examples $(\mathbf{x}_i, y_i)$, define $\mathbf{w}_n^* = \arg\min_{\mathbf{w}} \mathcal{P}_n(\mathbf{w})$ and compute $\mathbf{w}_n$ by applying the second-order SGD update (4) *to each of the n examples. If they converge, both $n\mathbb{E}\left[\,\|\mathbf{w}_n - \mathbf{w}^*\|^2\right]$ and $n\mathbb{E}\left[\,\|\mathbf{w}_n^* - \mathbf{w}^*\|^2\right]$ converge to a same positive constant K when n increases.*

This result means that, asymptotically and on average, the parameter $\mathbf{w}_n$ obtained after one pass of second-order SGD is as close to the infinite training set solution $\mathbf{w}^*$ as the true optimum of the primal $\mathbf{w}_n^*$. Therefore, when the training set is large enough, we can expect that a single pass of second-order SGD (n iterations of (4)) optimizes the primal accurately enough to replicate the test error of the actual SVM solution.

When we replace the full second-order rescaling matrix $\mathbf{B} = \mathbf{H}^{-1}$ by a more computationally acceptable approximation, Theorem 1 indicates that we lose a constant factor k on the required number of iterations to reach that accuracy. In other words, we can expect to replicate the SVM test error after k passes over the randomly reshuffled training set.

On the other hand, a well chosen approximation of the rescaling matrix can save a large constant factor on the computation of the generic SGD update (2). The best training times are therefore obtained by carefully trading the quality of the approximation for sparse representations.

	Frequency	Loss
Special example:	$\frac{n}{\texttt{skip}}$	$\frac{\lambda\texttt{skip}}{2}\|\mathbf{w}\|^2$
Examples 1 to n:	1	$\ell(y_i\mathbf{w}^\top\mathbf{x}_i)$

Table 2: The regularization term in the primal cost can be viewed as an additional training example with an arbitrarily chosen frequency and a specific loss function.

2.4 More Speedup Opportunities

We have argued that carefully designed quasi-Newton techniques can save a constant factor on the training times. There are of course many other ways to save constant factors:

- *Exploiting the sparsity of the patterns* (see Section 3) can save a constant factor in the cost of each first-order iteration. The benefits are more limited in the second-order case, because the inverse Hessian matrix is usually not sparse.
- *Implementation details* (see Section 4) such as compiler technology or parallelization on a predetermined number of processors can also reduce the learning time by constant factors.

Such opportunities are often dismissed as engineering tricks. However they should be considered on an equal footing with quasi-Newton techniques. Constant factors matter regardless of their origin. The following two sections provide a detailed discussion of sparsity and implementation.

3. Scheduling Stochastic Updates to Exploit Sparsity

First-order SGD iterations can be made substantially faster when the patterns $\mathbf{x}_t$ are sparse. The first-order SGD update has the form

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t - \alpha_t\mathbf{w}_t - \beta_t\mathbf{x}_t\,, \tag{5}$$

where α_t and β_t are scalar coefficients. Subtracting $\beta_t\mathbf{x}_t$ from the parameter vector involves solely the nonzero coefficients of the pattern $\mathbf{x}_t$. On the other hand, subtracting $\alpha_t\mathbf{w}_t$ involves all d coefficients. A naive implementation of (5) would therefore spend most of the time processing this first term. Shalev-Shwartz et al. (2007) circumvent this problem by representing the parameter $\mathbf{w}_t$ as the product $s_t\mathbf{v}_t$ of a scalar and a vector. The update (5) can then be computed as $s_{t+1} = (1-\alpha_t)s_t$ and $\mathbf{v}_{t+1} = \mathbf{v}_t - \beta\mathbf{x}_t/s_{t+1}$ in time proportional to the number of nonzero coefficients in $\mathbf{x}_t$.

Although this simple approach works well for the first order SGD algorithm, it does not extend nicely to quasi-Newton SGD algorithms. A more general method consists of treating the regularization term in the primal cost (1) as an additional training example occurring with an arbitrarily chosen frequency with a specific loss function.

Consider examples with the frequencies and losses listed in Table 2 and write the average loss:

$$\frac{1}{\frac{n}{\texttt{skip}}+n}\left[\frac{n}{\texttt{skip}}\left(\frac{\lambda\texttt{skip}}{2}\|\mathbf{w}\|^2\right) \;+\; \sum_{i=1}^{n}\ell(y_i\mathbf{w}^\top\mathbf{x}_i)\right] = \frac{\texttt{skip}}{1+\texttt{skip}}\left[\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\ell(y_i\mathbf{w}^\top\mathbf{x}_i)\right].$$

SGD	SVMSGD2
Require: $\lambda, \mathbf{w}_0, t_0, T$	**Require:** $\lambda, \mathbf{w}_0, t_0, T$, `skip`
1: $t = 0$	1: $t = 0$, `count` $=$ `skip`
2: **while** $t \leq T$ **do**	2: **while** $t \leq T$ **do**
3: $\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{\lambda(t+t_0)}(\lambda \mathbf{w}_t + \ell'(y_t \mathbf{w}_t^\top \mathbf{x}_t) y_t \mathbf{x}_t)$	3: $\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{\lambda(t+t_0)}\ell'(y_t \mathbf{w}_t^\top \mathbf{x}_t) y_t \mathbf{x}_t$
4:	4: `count` $=$ `count` -1
5:	5: **if** `count` ≤ 0 **then**
6:	6: $\mathbf{w}_{t+1} = \mathbf{w}_{t+1} - \frac{\texttt{skip}}{t+t_0}\mathbf{w}_{t+1}$
7:	7: `count` $=$ `skip`
8:	8: **end if**
9: $t = t + 1$	9: $t = t + 1$
10: **end while**	10: **end while**
11: **return** $\mathbf{w}_T$	11: **return** $\mathbf{w}_T$

Figure 1: Detailed pseudo-codes of the SGD and SVMSGD2 algorithms.

Minimizing this loss is of course equivalent to minimizing the primal cost (1) with its regularization term. Applying the SGD algorithm to the examples defined in Table 2 separates the regularization updates, which involve the special example, from the pattern updates, which involve the real examples. The parameter `skip` regulates the relative frequencies of these updates. The SVMSGD2 algorithm (Bottou, 2007) measures the average pattern sparsity and picks a frequency that ensures that the amortized cost of the regularization update is proportional to the number of nonzero coefficients. Figure 1 compares the pseudo-codes of the naive first-order SGD and of the first-order SVMSGD2. Both algorithms handle the real examples at each iteration (line 3) but SVMSGD2 only performs a regularization update every `skip` iterations (line 6).

Assume s is the average proportion of nonzero coefficients in the patterns $\mathbf{x}_i$ and set `skip` to c/s where c is a predefined constant (we use $c = 16$ in our experiments). Each pattern update (line 3) requires sd operations. Each regularization update (line 6) requires d operations but occurs s/c times less often. The average cost per iteration is therefore proportional to $O(sd)$ instead of $O(d)$.

4. Implementation

In the optimization literature, a superior algorithm implemented with a slow scripting language usually beats careful implementations of inferior algorithms. This is because the superior algorithm minimizes the training error with a higher order convergence.

This is no longer true in the case of large scale machine learning because we care about the test error instead of the training error. As explained above, algorithm improvements do not improve the order of the test error convergence. They can simply improve constant factors and therefore *compete evenly with implementation improvements*. Time spent refining the implementation is time well spent.

- There are lots of methods for representing sparse vectors with sharply different computing requirement for sequential and random access. Our C++ implementation always uses a full

	Full	**Sparse**
Random access to a single coefficient:	$\Theta(1)$	$\Theta(s)$
In-place addition into a full vector of dimension d':	$\Theta(d)$	$\Theta(s)$
In-place addition into a sparse vector with s' nonzeros:	$\Theta(d+s')$	$\Theta(s+s')$

Table 3: Costs of various operations on a vector of dimension d with s nonzero coefficients.

vector for representing the parameter $\mathbf{w}$, but handles the patterns $\mathbf{x}$ using either a full vector representation or a sparse representation as an ordered list of index/value pairs.

Each calculation can be achieved directly on the sparse representation or after a conversion to the full representation (see Table 3). Inappropriate choices have outrageous costs. For example, on a dense data set with 500 attributes, using sparse vectors increases the training time by 50%; on the sparse RCV1 data set (see Table 4), using a sparse vector to represent the parameter $\mathbf{w}$ increases the training time by more than 900%.

- Modern processors often sport specialized instructions to handle vectors and multiple cores. Linear algebra libraries, such as `BLAS`, may or may not use them in ways that suit our purposes. Compilation flags have nontrivial impacts on the learning times.

Such implementation improvements are often (but not always) orthogonal to the algorithmic improvements described above. The main issue consists of deciding how much development resources are allocated to implementation and to algorithm design. This trade-off depends on the available competencies.

5. SGD-QN: A Careful Diagonal Quasi-Newton SGD

As explained in Section 2, designing an efficient quasi-Newton SGD algorithm involves a careful trade-off between the sparsity of the scaling matrix representation $\mathbf{B}$ and the quality of its approximation of the inverse Hessian $\mathbf{H}^{-1}$. The two obvious choices are diagonal approximations (Becker and Le Cun, 1989) and low rank approximations (Schraudolph et al., 2007).

5.1 Diagonal Rescaling Matrices

Among numerous practical suggestions for running SGD algorithm in multilayer neural networks, Le Cun et al. (1998) emphatically recommend to rescale each input space feature in order to improve the condition number κ of the Hessian matrix. In the case of a linear model, such preconditioning is similar to using a constant diagonal scaling matrix.

Rescaling the input space defines transformed patterns $\mathbf{X}_t$ such that $[\mathbf{X}_t]_i = b_i [\mathbf{x}_t]_i$ where the notation $[\mathbf{v}]_i$ represents the i-th coefficient of vector $\mathbf{v}$. This transformation does not change the classification if the parameter vectors are modified as $[\mathbf{W}_t]_i = [\mathbf{w}_t]_i / b_i$. The first-order SGD update on these modified variable is then

$$\begin{aligned} \forall i = 1 \dots d \qquad [\mathbf{W}_{t+1}]_i &= [\mathbf{W}_t]_i - \eta_t \left(\lambda [\mathbf{W}_t]_i + \ell'(y_t \mathbf{W}_t^\top \mathbf{X}_t)\, y_t\, [\mathbf{X}_t]_i, \right) \\ &= [\mathbf{W}_t]_i - \eta_t \left(\lambda [\mathbf{W}_t]_i + \ell'(y_t \mathbf{w}_t^\top \mathbf{x}_t)\, y_t\, b_i [\mathbf{x}_t]_i \right) . \end{aligned}$$

Multiplying by b_i shows how the original parameter vector $\mathbf{w}_t$ is affected:

$$\forall i = 1 \dots d \qquad [\mathbf{w}_{t+1}]_i = [\mathbf{w}_t]_i - \eta_t \left(\lambda [\mathbf{w}_t]_i + \ell'(y_t \mathbf{w}_t^\top \mathbf{x}_t) \, y_t \, b_i^2 \, [\mathbf{x}_t]_i \right) .$$

We observe that rescaling the input is equivalent to multiplying the gradient by a fixed diagonal matrix $\mathbf{B}$ whose elements are the squares of the coefficients b_i.

Ideally we would like to make the product $\mathbf{BH}$ spectrally close the identity matrix. Unfortunately we do not know the value of the Hessian matrix $\mathbf{H}$ at the optimum $\mathbf{w}_n^*$. Instead we could consider the current value of the Hessian $\mathbf{H}_{\mathbf{w}_t} = \mathcal{P}''(\mathbf{w}_t)$ and compute the diagonal rescaling matrix $\mathbf{B}$ that makes $\mathbf{BH}_{\mathbf{w}_t}$ closest to the identity. This computation could be very costly because it involves the full Hessian matrix. Becker and Le Cun (1989) approximate the optimal diagonal rescaling matrix by inverting the diagonal coefficients of the Hessian. The method relies on the analytical derivation of these diagonal coefficients for multilayer neural networks. This derivation does not extend to arbitrary models. It certainly does not work in the case of traditional SVMs because the hinge loss has zero curvature almost everywhere.

5.2 Low Rank Rescaling Matrices

The popular LBFGS optimization algorithm (Nocedal, 1980) maintains a low rank approximation of the inverse Hessian by storing the k most recent rank-one BFGS updates instead of the full inverse Hessian matrix. When the successive full gradients $\mathcal{P}_n'(\mathbf{w}_{t-1})$ and $\mathcal{P}_n'(\mathbf{w}_t)$ are available, standard rank one updates can be used to directly estimate the inverse Hessian matrix $\mathbf{H}^{-1}$. Using this method with stochastic gradient is tricky because the full gradients $\mathcal{P}_n'(\mathbf{w}_{t-1})$ and $\mathcal{P}_n'(\mathbf{w}_t)$ are not readily available. Instead we only have access to the stochastic estimates $\mathbf{g}_{t-1}(\mathbf{w}_{t-1})$ and $\mathbf{g}_t(\mathbf{w}_t)$ which are too noisy to compute good rescaling matrices.

The oLBFGS algorithm (Schraudolph et al., 2007) compares instead the derivatives $\mathbf{g}_{t-1}(\mathbf{w}_{t-1})$ and $\mathbf{g}_{t-1}(\mathbf{w}_t)$ for the same example $(\mathbf{x}_{t-1}, y_{t-1})$. This reduces the noise to an acceptable level at the expense of the computation of the additional gradient vector $\mathbf{g}_{t-1}(\mathbf{w}_t)$.

Compared to the first-order SGD, each iteration of the oLBFGS algorithm computes the additional quantity $\mathbf{g}_{t-1}(\mathbf{w}_t)$ and updates the list of k rank one updates. The most expensive part however remains the multiplication of the gradient $\mathbf{g}_t(\mathbf{w}_t)$ by the low-rank estimate of the inverse Hessian. With $k = 10$, each iteration of our oLBFGS implementation runs empirically 11 times slower than a first-order SGD iteration.

5.3 SGD-QN

The SGD-QN algorithm estimates a *diagonal rescaling matrix* using a technique inspired by oLBFGS. For any pair of parameters $\mathbf{w}_{t-1}$ and $\mathbf{w}_t$, a Taylor series of the gradient of the primal cost $\mathcal{P}$ provides the secant equation:

$$\mathbf{w}_t - \mathbf{w}_{t-1} \approx \mathbf{H}_{\mathbf{w}_t}^{-1} \left(\mathcal{P}_n'(\mathbf{w}_t) - \mathcal{P}_n'(\mathbf{w}_{t-1}) \right) . \tag{6}$$

We would then like to replace the inverse Hessian matrix $\mathbf{H}_{\mathbf{w}_t}^{-1}$ by a diagonal estimate $\mathbf{B}$

$$\mathbf{w}_t - \mathbf{w}_{t-1} \approx \mathbf{B} \left(\mathcal{P}_n'(\mathbf{w}_t) - \mathcal{P}_n'(\mathbf{w}_{t-1}) \right) .$$

Since we are designing a stochastic algorithm, we do not have access to the full gradient $\mathcal{P}_n'$. Following oLBFGS, we replace them by the local gradients $\mathbf{g}_{t-1}(\mathbf{w}_t)$ and $\mathbf{g}_{t-1}(\mathbf{w}_{t-1})$ and obtain

$$\mathbf{w}_t - \mathbf{w}_{t-1} \approx \mathbf{B} \left(\mathbf{g}_{t-1}(\mathbf{w}_t) - \mathbf{g}_{t-1}(\mathbf{w}_{t-1}) \right) .$$

Since we chose to use a diagonal rescaling matrix $\mathbf{B}$, we can write the term-by-term equality

$$[\mathbf{w}_t - \mathbf{w}_{t-1}]_i \approx \mathbf{B}_{ii} [\mathbf{g}_{t-1}(\mathbf{w}_t) - \mathbf{g}_{t-1}(\mathbf{w}_{t-1})]_i ,$$

where the notation $[\mathbf{v}]_i$ still represents the i-th coefficient of vector $\mathbf{v}$. This leads to computing $\mathbf{B}_{ii}$ as the average of the ratio $[\mathbf{w}_t - \mathbf{w}_{t-1}]_i / [\mathbf{g}_{t-1}(\mathbf{w}_t) - \mathbf{g}_{t-1}(\mathbf{w}_{t-1})]_i$. An online estimation is easily achieved during the course of learning by performing a leaky average of these ratios,

$$\mathbf{B}_{ii} \leftarrow \mathbf{B}_{ii} + \frac{2}{r} \left(\frac{[\mathbf{w}_t - \mathbf{w}_{t-1}]_i}{[\mathbf{g}_{t-1}(\mathbf{w}_t) - \mathbf{g}_{t-1}(\mathbf{w}_{t-1})]_i} - \mathbf{B}_{ii} \right) \qquad \forall i = 1 \ldots d ,$$

and where the integer r is incremented whenever we update the matrix $\mathbf{B}$.

The weights of the scaling matrix $\mathbf{B}$ are initialized to λ^{-1} because this corresponds to the exact setup of first-order SGD. Since the curvature of the primal cost (1) is always larger than λ, the ratio $[\mathbf{g}_{t-1}(\mathbf{w}_t) - \mathbf{g}_{t-1}(\mathbf{w}_{t-1})]_i / [\mathbf{w}_t - \mathbf{w}_{t-1}]_i$ is always larger than λ. Therefore the coefficients $\mathbf{B}_{ii}$ never exceed their initial value λ^{-1}. Basically these scaling factors slow down the convergence along some axes. The speedup does not occur because we follow the trajectory faster, but because we follow a better trajectory.

Performing the weight update (2) with a diagonal rescaling matrix $\mathbf{B}$ consists in performing term-by-term operations with a time complexity that is marginally greater than the complexity of the first-order SGD (3) update. The computation of the additional gradient vector $\mathbf{g}_{t-1}(\mathbf{w}_t)$ and the reestimation of all the coefficients $\mathbf{B}_{ii}$ essentially triples the computing time of a first-order SGD iteration with non-sparse inputs (3), and is considerably slower than a first-order SGD iteration with sparse inputs implemented as discussed in Section 3.

Fortunately this higher computational cost per iteration can be nearly avoided by scheduling the reestimation of the rescaling matrix with the same frequency as the regularization updates. Section 5.1 has shown that a diagonal rescaling matrix does little more than rescaling the input variables. Since a fixed diagonal rescaling matrix already works quite well, there is little need to update its coefficients very often.

Figure 2 compares the SVMSGD2 and SGD-QN algorithms. Whenever SVMSGD2 performs a regularization update, we set the flag `updateB` to schedule a reestimation of the rescaling coefficients during the next iteration. This is appropriate because both operations have comparable computing times. Therefore the rescaling matrix reestimation schedule can be regulated with the same `skip` parameter as the regularization updates. In practice, we observe that each SGD-QN iteration demands less than twice the time of a first-order SGD iteration.

Because SGD-QN reestimates the rescaling matrix after a pattern update, special care must be taken when the ratio $[\mathbf{w}_t - \mathbf{w}_{t-1}]_i / [\mathbf{g}_{t-1}(\mathbf{w}_t) - \mathbf{g}_{t-1}(\mathbf{w}_{t-1})]_i$ has the form $0/0$ because the corresponding input coefficient $[\mathbf{x}_{t-1}]_i$ is zero. Since the secant Equation (6) is valid for any two values of the parameter vector, one can compute the ratios with parameter vectors $\mathbf{w}_{t-1}$ and $\mathbf{w}_t + \varepsilon$ and derive the correct value by continuity when $\varepsilon \to 0$. When $[\mathbf{x}_{t-1}]_i = 0$, we can write

$$\begin{aligned}
\frac{[(\mathbf{w}_t+\varepsilon)-\mathbf{w}_{t-1}]_i}{[\mathbf{g}_{t-1}(\mathbf{w}_t+\varepsilon)-\mathbf{g}_{t-1}(\mathbf{w}_{t-1})]_i} &= \frac{[(\mathbf{w}_t+\varepsilon)-\mathbf{w}_{t-1}]_i}{\lambda[(\mathbf{w}_t+\varepsilon)-\mathbf{w}_{t-1}]_i + \left(\ell'(y_{t-1}(\mathbf{w}_t+\varepsilon)^\top \mathbf{x}_{t-1}) - \ell'(y_{t-1}\mathbf{w}_{t-1}^\top \mathbf{x}_{t-1})\right) y_{t-1} [\mathbf{x}_{t-1}]_i} \\
&= \left(\lambda + \frac{\left(\ell'(y_{t-1}(\mathbf{w}_t+\varepsilon)^\top \mathbf{x}_{t-1}) - \ell'(y_{t-1}\mathbf{w}_{t-1}^\top \mathbf{x}_{t-1})\right) y_{t-1} [\mathbf{x}_{t-1}]_i}{[(\mathbf{w}_t+\varepsilon)-\mathbf{w}_{t-1}]_i} \right)^{-1} \\
&= \left(\lambda + \frac{0}{[\varepsilon]_i} \right)^{-1} \quad \stackrel{\varepsilon \to 0}{\longrightarrow} \quad \lambda^{-1} .
\end{aligned}$$

SVMSGD2	SGD-QN
Require: λ, $\mathbf{w}_0$, t_0, T, skip	**Require:** λ, $\mathbf{w}_0$, t_0, T, skip
$t = 0$, count = skip	$t = 0$, count = skip,
	$\mathbf{B} = \lambda^{-1}\mathbf{I}$; updateB = false; $r = 2$
while $t \leq T$ **do**	**while** $t \leq T$ **do**
$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{\lambda(t+t_0)}\ell'(y_t\mathbf{w}_t^\top\mathbf{x}_t)y_t\mathbf{x}_t$	$\mathbf{w}_{t+1} = \mathbf{w}_t - (t+t_0)^{-1}\ell'(y_t\mathbf{w}_t^\top\mathbf{x}_t)y_t\,\mathbf{B}\,\mathbf{x}_t$
	if updateB = true **then**
	$\mathbf{p}_t = \mathbf{g}_t(\mathbf{w}_{t+1}) - \mathbf{g}_t(\mathbf{w}_t)$
	$\forall i, \mathbf{B}_{ii} = \mathbf{B}_{ii} + \frac{2}{r}\left([\mathbf{w}_{t+1} - \mathbf{w}_t]_i\,[\mathbf{p}_t]_i^{-1} - \mathbf{B}_{ii}\right)$
	$\forall i, \mathbf{B}_{ii} = \max(\mathbf{B}_{ii}, 10^{-2}\lambda^{-1})$
	$r = r + 1$; updateB = false
	end if
count = count-1	count = count-1
if count ≤ 0 **then**	**if** count ≤ 0 **then**
$\mathbf{w}_{t+1} = \mathbf{w}_{t+1} - \text{skip}\,(t+t_0)^{-1}\mathbf{w}_{t+1}$	$\mathbf{w}_{t+1} = \mathbf{w}_{t+1} - \text{skip}\,(t+t_0)^{-1}\lambda\,\mathbf{B}\,\mathbf{w}_{t+1}$
count = skip	count = skip; updateB = true
end if	**end if**
$t = t + 1$	$t = t + 1$
end while	**end while**
return $\mathbf{w}_T$	**return** $\mathbf{w}_T$

Figure 2: Detailed pseudo-codes of the SVMSGD2 and SGD-QN algorithms.

Data Set	Train. Ex.	Test. Ex.	Features	s	λ	t_0	*skip*
ALPHA	100,000	50,000	500	1	10^{-5}	10^6	16
DELTA	100,000	50,000	500	1	10^{-4}	10^4	16
RCV1	781,265	23,149	47,152	0.0016	10^{-4}	10^5	9,965

Table 4: Data sets and parameters used for experiments.

6. Experiments

We demonstrate the good scaling properties of SGD-QN in two ways: we present a detailed comparison with other stochastic gradient methods, and we summarize the results obtained on the PASCAL Large Scale Challenge.

Table 4 describes the three binary classification tasks we used for comparative experiments. The Alpha and Delta tasks were defined for the PASCAL Large Scale Challenge (Sonnenburg et al., 2008). We train with the first 100,000 examples and test with the last 50,000 examples of the official training sets because the official testing sets are not available. Alpha and Delta are dense data sets with relatively severe conditioning problems. The third task is the classification of RCV1 documents belonging to class CCAT (Lewis et al., 2004). This task has become a standard benchmark for linear SVMs on sparse data. Despite its larger size, the RCV1 task is much easier than the Alpha and Delta tasks. All methods discussed in this paper perform well on RCV1.

	ALPHA	RCV1
SGD	0.13	36.8
SVMSGD2	0.10	0.20
SGD-QN	0.21	0.37

Table 5: Time (sec.) for performing one pass over the training set.

The experiments reported in Section 6.4 use the hinge loss $\ell(s) = \max(0, 1-s)$. All other experiments use the squared hinge loss $\ell(s) = \frac{1}{2}(\max(0, 1-s))^2$. In practice, there is no need to make the losses twice differentiable by smoothing their behavior near $s = 0$. Unlike most batch optimizer, stochastic algorithms do not not aim directly for nondifferentiable points, but randomly hop around them. The stochastic noise implicitly smoothes the loss.

The SGD, SVMSGD2, oLBFGS, and SGD-QN algorithms were implemented using the same C++ code base.[3] All experiments are carried out in single precision. We did not experience numerical accuracy issues, probably because of the influence of the regularization term. Our implementation of oLBFGS maintains a rank 10 rescaling matrix. Setting the oLBFGS gain schedule is rather delicate. We obtained fairly good results by replicating the gain schedule of the VieCRF package.[4] We also propose a comparison with the online dual linear SVM solver (Hsieh et al., 2008) implemented in the LibLinear package.[5] We did not reimplement this algorithm because the LibLinear implementation has proved as simple and as efficient as ours.

The t_0 parameter is determined using an automatic procedure: since the size of the training set does not affect results of Theorem 1, we simply pick a subset containing 10% of the training examples, perform one SGD-QN pass over this subset with several values for t_0, and pick the value for which the primal cost decreases the most. These values are given in Table 4.

6.1 Sparsity Tricks

Table 5 illustrates the influence of the scheduling tricks described in Section 3. The table displays the training times of SGD and SVMSGD2. The only difference between these two algorithms are the scheduling tricks. SVMSGD2 trains 180 times faster than SGD on the sparse data set RCV1. This table also demonstrates that iterations of the quasi-newton SGD-QN are not prohibitively expensive.

6.2 Quasi-Newton

Figure 3 shows how the primal cost $\mathcal{P}_n(\mathbf{w})$ of the Alpha data set evolves with the number of passes (left) and the training time (right). Compared to the first-order SVMSGD2, both the oLBFGS and SGD-QN algorithms dramatically decrease the number of passes required to achieve similar values of the primal. Even if it uses a more precise approximation of the inverse Hessian, oLBFGS does not perform better after a single pass than SGD-QN. Besides, running a single pass of oLBFGS is much slower than running multiple passes of SVMSGD2 or SGD-QN. The benefits of its second-order approximation are canceled by its greater time requirements per iteration. On the other hand,

3. Implementations and experiment scripts are available in the `libsgdqn` library on `http://www.mloss.org`.
4. This can be found at `http://www.ofai.at/~jeremy.jancsary`.
5. This can be found at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

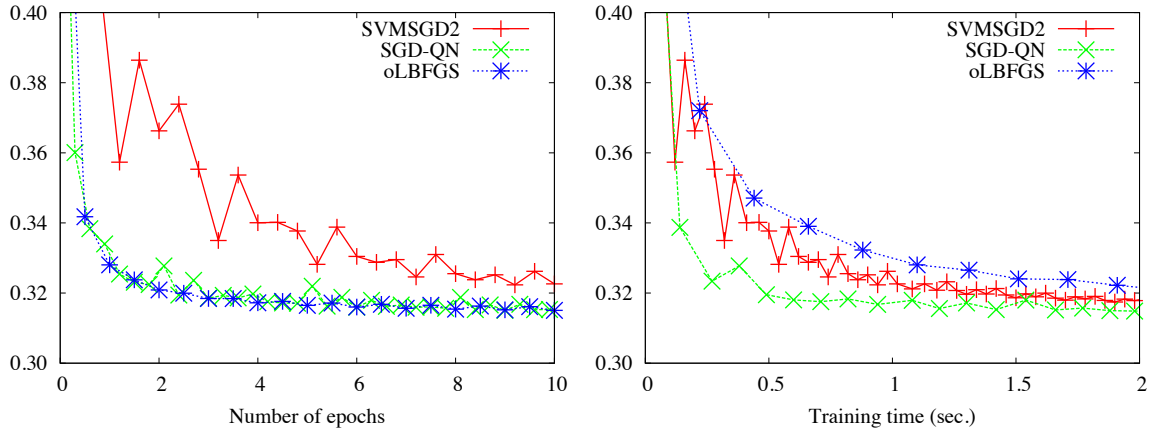


Figure 3: Primal costs according to the number of epochs (left) and the training duration (right) on the Alpha data set.

each SGD-QN iteration is only marginally slower than a SVMSGD2 iteration; the reduction of the number of iterations is sufficient to offset this cost.

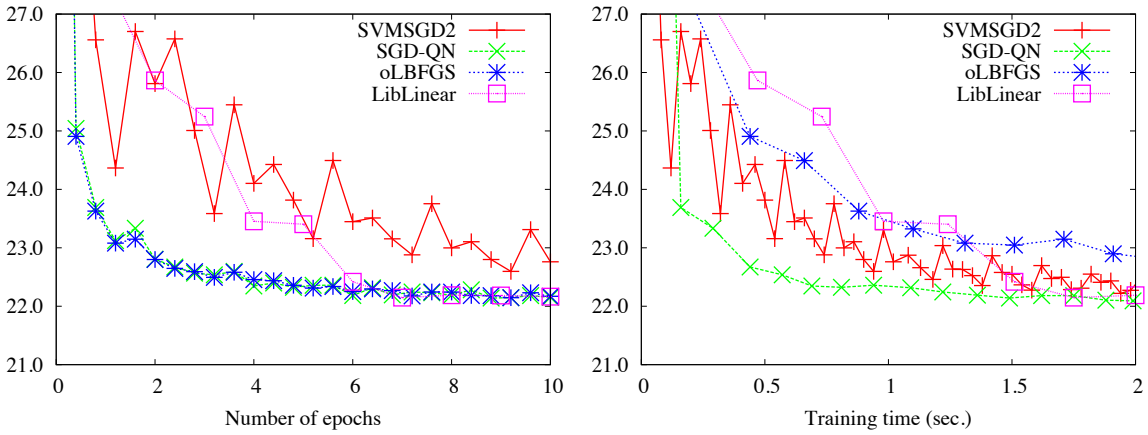
6.3 Training Speed

Figure 4 displays the test errors achieved on the Alpha, Delta and RCV1 data sets as a function of the number of passes (left) and the training time (right). These results show again that both oLBFGS and SGD-QN require less iterations than SVMSGD2 to achieve the same test error. However, oLBFGS suffers from the relatively high complexity of its update process. The SGD-QN algorithm is competitive with the dual solver LibLinear on the dense data sets Alpha and Delta; it runs significantly faster on the sparse RCV1 data set.

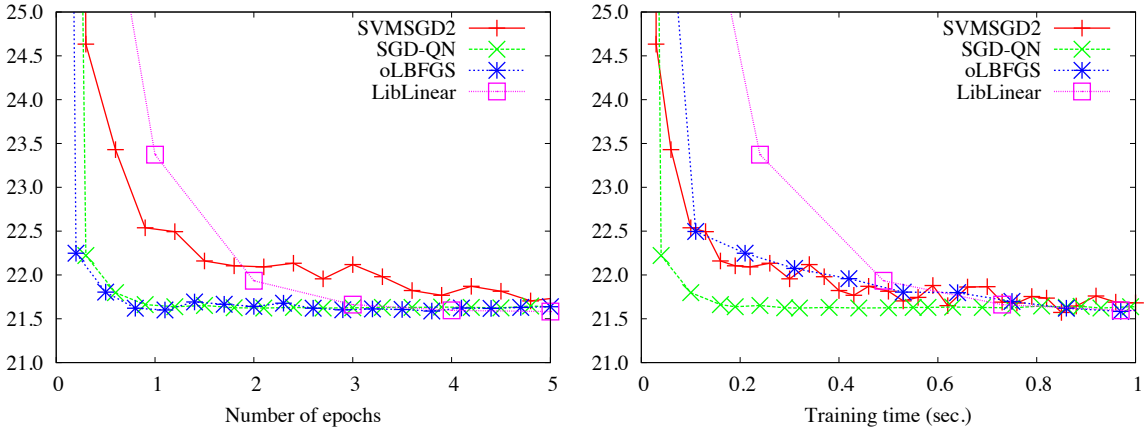
According to Theorem 4, given a large enough training set, a perfect second-order SGD algorithm would reach the batch test error after a single pass. One pass learning is attractive when we are dealing with high volume streams of examples that cannot be stored and retrieved quickly. Figure 4 (left) shows that oLBFGS is a little bit closer to that ideal than SGD-QN and could become attractive for problems where the example retrieval time is much greater than the computing time.

6.4 PASCAL Large Scale Challenge Results

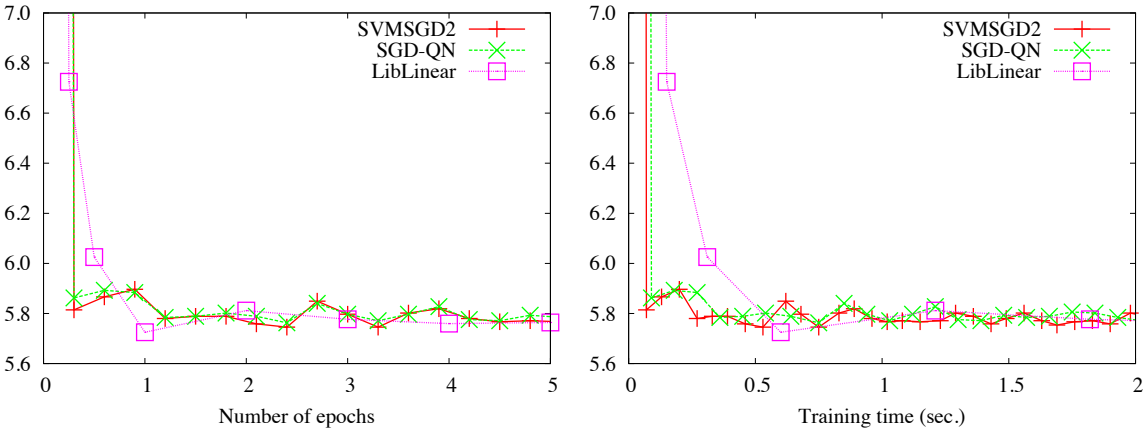
The SGD-QN algorithm has been submitted to the "Wild Track" of the PASCAL Large Scale Challenge. Wild Track contributors were free to do anything leading to more efficient and more accurate methods. Forty two methods have been submitted to this track. Table 6 shows the SGD-QN ranks determined by the organizers of the challenge according to their evaluation criteria. The SGD-QN algorithm always ranks among the top five submissions and ranks first in overall score (tie with another Newton method).



ALPHA DATA SET



DELTA DATA SET



RCV1 DATA SET

Figure 4: Test errors (in %) according to the number of epochs (left) and training duration (right).

Data Set	λ	*skip*	Passes	**Rank**
Alpha	10^{-5}	16	10	$\mathbf{1}^{st}$
Beta	10^{-4}	16	15	$\mathbf{3}^{rd}$
Gamma	10^{-3}	16	10	$\mathbf{1}^{st}$
Delta	10^{-3}	16	10	$\mathbf{1}^{st}$
Epsilon	10^{-5}	16	10	$\mathbf{5}^{th}$
Zeta	10^{-5}	16	10	$\mathbf{4}^{th}$
OCR	10^{-5}	16	10	$\mathbf{2}^{nd}$
Face	10^{-5}	16	20	$\mathbf{4}^{th}$
DNA	10^{-3}	64	10	$\mathbf{2}^{nd}$
Webspam	10^{-5}	71,066	10	$\mathbf{4}^{th}$

Table 6: Parameters and final ranks obtained by SGD-QN in the "Wild Track" of the first PASCAL Large Scale Learning Challenge. All competing algorithms were run by the organizers. (Note: the competition results were obtained with a preliminary version of SGD-QN. In particular the λ parameters listed above are different from the values used for all experiments in this paper and listed in Table 4.)

7. Conclusion

The SGD-QN algorithm strikes a good compromise for large scale application because it has low time and memory requirements per iteration and because it reaches competitive test errors after a small number of iterations. We have shown how this performance is the result of a careful design taking into account the theoretical knowledge about second-order SGD and a precise understanding of its computational requirements.

Finally, although this contribution presents SGD-QN as a solver for linear SVMs, this algorithm can be easily extended to nonlinear models for which we can analytically compute the gradients. We plan to further investigate the performance of SGD-QN in this context.

Acknowledgments

Part of this work was funded by NSF grant CCR-0325463 and by the EU Network of Excellence PASCAL2. Antoine Bordes was also supported by the French DGA.

Appendix A. Proof of Theorem 1

Define $\mathbf{v}_t = \mathbf{w}_t - \mathbf{w}_n^*$ and observe that a second-order Taylor expansion of the primal gives

$$\mathcal{P}_n(\mathbf{w}_t) - \mathcal{P}_n(\mathbf{w}_n^*) \;=\; \mathbf{v}_t^\top \mathbf{H} \mathbf{v}_t + \mathrm{o}\left(t^{-2}\right) \;=\; \mathbf{tr}\left(\mathbf{H}\mathbf{v}_t\mathbf{v}_t^\top\right) + \mathrm{o}\left(t^{-2}\right).$$

Let $\mathbb{E}_{t-1}$ representing the conditional expectation over the choice of the example at iteration $t-1$ given all the choices made during the previous iterations. Since we assume that convergence takes

place, we have

$$\begin{aligned}\mathbb{E}_{t-1}\left[\mathbf{g}_{t-1}(\mathbf{w}_{t-1})\,\mathbf{g}_{t-1}(\mathbf{w}_{t-1})^\top\right] &= \mathbb{E}_{t-1}\left[\mathbf{g}_{t-1}(\mathbf{w}_n^*)\,\mathbf{g}_{t-1}(\mathbf{w}_n^*)^\top\right] + \mathrm{o}(1) \;=\; \mathbf{G} + \mathrm{o}(1)\\ \text{and } \mathbb{E}_{t-1}\left[\mathbf{g}_{t-1}(\mathbf{w}_{t-1})\right] &= \mathcal{P}_n'(\mathbf{w}_{t-1}) \;=\; \mathbf{H}\mathbf{v}_{t-1} + \mathrm{o}(\mathbf{v}_{t-1}) \;=\; \mathbf{I}_\varepsilon \mathbf{H}\mathbf{v}_{t-1}\end{aligned}$$

where notation $\mathbf{I}_\varepsilon$ is a shorthand for $\mathbf{I} + \mathrm{o}(1)$, that is, a matrix that converges to the identity. Expressing $\mathbf{H}\mathbf{v}_t\mathbf{v}_t^\top$ using the generic SGD update (2) gives

$$\begin{aligned}\mathbf{H}\mathbf{v}_t\mathbf{v}_t^\top &= \mathbf{H}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^\top - \frac{\mathbf{H}\mathbf{v}_{t-1}\,\mathbf{g}_{t-1}(\mathbf{w}_{t-1})^\top\mathbf{B}}{t+t_0} - \frac{\mathbf{H}\mathbf{B}\,\mathbf{g}_{t-1}(\mathbf{w}_{t-1})\,\mathbf{v}_{t-1}^\top}{t+t_0}\\ &\quad + \frac{\mathbf{H}\mathbf{B}\,\mathbf{g}_{t-1}(\mathbf{w}_{t-1})\,\mathbf{g}_{t-1}(\mathbf{w}_{t-1})^\top\mathbf{B}}{(t+t_0)^2}\\ \mathbb{E}_{t-1}\left[\mathbf{H}\mathbf{v}_t\mathbf{v}_t^\top\right] &= \mathbf{H}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^\top - \frac{\mathbf{H}\mathbf{v}_{t-1}\,\mathbf{v}_{t-1}^\top\mathbf{H}\mathbf{I}_\varepsilon\mathbf{B}}{t+t_0} - \frac{\mathbf{H}\mathbf{B}\mathbf{I}_\varepsilon\mathbf{H}\mathbf{v}_{t-1}\,\mathbf{v}_{t-1}^\top}{t+t_0} + \frac{\mathbf{H}\mathbf{B}\mathbf{G}\mathbf{B}}{(t+t_0)^2} + \mathrm{o}\left(t^{-2}\right)\\ \mathbb{E}_{t-1}\left[\mathbf{tr}\left(\mathbf{H}\mathbf{v}_t\mathbf{v}_t^\top\right)\right] &= \mathbf{tr}\left(\mathbf{H}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^\top\right) - \frac{2\,\mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{I}_\varepsilon\mathbf{H}\mathbf{v}_{t-1}\,\mathbf{v}_{t-1}^\top\right)}{t+t_0} + \frac{\mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{G}\mathbf{B}\right)}{(t+t_0)^2} + \mathrm{o}\left(t^{-2}\right)\\ \mathbb{E}_\sigma\left[\mathbf{tr}\left(\mathbf{H}\mathbf{v}_t\mathbf{v}_t^\top\right)\right] &= \mathbb{E}_\sigma\left[\mathbf{tr}\left(\mathbf{H}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^\top\right)\right] - \frac{2\mathbb{E}_\sigma\left[\mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{I}_\varepsilon\mathbf{H}\mathbf{v}_{t-1}\,\mathbf{v}_{t-1}^\top\right)\right]}{t+t_0} + \frac{\mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{G}\mathbf{B}\right)}{(t+t_0)^2} + \mathrm{o}\left(t^{-2}\right).\end{aligned}$$

Let $\lambda_{\max} \geq \lambda_{\min} > 1/2$ be the extreme eigenvalues of $\mathbf{HB}$. Since, for any positive matrix $\mathbf{X}$,

$$\left(\lambda_{\min} + \mathrm{o}(1)\right)\mathbf{tr}\left(\mathbf{X}\right) \;\leq\; \mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{I}_\varepsilon\mathbf{X}\right) \;\leq\; \left(\lambda_{\max} + \mathrm{o}(1)\right)\mathbf{tr}\left(\mathbf{X}\right)$$

we can bracket $\mathbb{E}_\sigma\left[\mathbf{tr}\left(\mathbf{H}\mathbf{v}_t\mathbf{v}_t^\top\right)\right]$ between the expressions

$$\left(1 - \frac{2\lambda_{\max}}{t} + \mathrm{o}\left(\frac{1}{t}\right)\right)\mathbb{E}_\sigma\left[\mathbf{tr}\left(\mathbf{H}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^\top\right)\right] + \frac{\mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{G}\mathbf{B}\right)}{(t+t_0)^2} + \mathrm{o}\left(t^{-2}\right)$$

and

$$\left(1 - \frac{2\lambda_{\min}}{t} + \mathrm{o}\left(\frac{1}{t}\right)\right)\mathbb{E}_\sigma\left[\mathbf{tr}\left(\mathbf{H}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^\top\right)\right] + \frac{\mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{G}\mathbf{B}\right)}{(t+t_0)^2} + \mathrm{o}\left(t^{-2}\right)$$

By recursively applying this bracket, we obtain

$$u_{\lambda_{\max}}(t+t_0) \;\leq\; \mathbb{E}_\sigma\left[\mathbf{tr}\left(\mathbf{H}\mathbf{v}_t\mathbf{v}_t^\top\right)\right] \;\leq\; u_{\lambda_{\min}}(t+t_0)$$

where the notation $u_\lambda(t)$ represents a sequence of real satisfying the recursive relation

$$u_\lambda(t) \;=\; \left(1 - \frac{2\lambda}{t} + \mathrm{o}\left(\frac{1}{t}\right)\right)u_\lambda(t-1) + \frac{\mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{G}\mathbf{B}\right)}{t^2} + \mathrm{o}\left(\frac{1}{t^2}\right).$$

From (Bottou and LeCun, 2005, Lemma 1), $\lambda > 1/2$ implies $t\,u_\lambda(t) \longrightarrow \frac{\mathbf{tr}(\mathbf{HBGB})}{2\lambda-1}$. Then

$$\frac{\mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{G}\mathbf{B}\right)}{2\lambda_{\max}-1}t^{-1} + \mathrm{o}\left(t^{-1}\right) \leq \mathbb{E}_\sigma\left[\mathbf{tr}\left(\mathbf{H}\mathbf{v}_t\mathbf{v}_t^\top\right)\right] \leq \frac{\mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{G}\mathbf{B}\right)}{2\lambda_{\min}-1}t^{-1} + \mathrm{o}\left(t^{-1}\right)$$

and

$$\frac{\mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{G}\mathbf{B}\right)}{2\lambda_{\max}-1}t^{-1} + \mathrm{o}\left(t^{-1}\right) \leq \mathbb{E}_\sigma\left[\mathcal{P}_n(\mathbf{w}_t) - \mathcal{P}_n(\mathbf{w}_n^*)\right] \leq \frac{\mathbf{tr}\left(\mathbf{H}\mathbf{B}\mathbf{G}\mathbf{B}\right)}{2\lambda_{\min}-1}t^{-1} + \mathrm{o}\left(t^{-1}\right). \qquad \blacksquare$$

References

S.-I. Amari, H. Park, and K. Fukumizu. Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, 12:1409, 2000.

S. Becker and Y. Le Cun. Improving the convergence of back-propagation learning with second-order methods. In *Proc. 1988 Connectionist Models Summer School*, pages 29–37. Morgan Kaufman, 1989.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *J. Machine Learning Research*, 6:1579–1619, September 2005.

L. Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998.

L. Bottou. Stochastic gradient descent on toy problems, 2007. `http://leon.bottou.org/projects/sgd`.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Adv. in Neural Information Processing Systems*, volume 20. MIT Press, 2008.

L. Bottou and Y. LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.

X. Driancourt. *Optimisation par descente de gradient stochastique de systèmes modulaires combinant réseaux de neurones et programmation dynamique*. PhD thesis, Université Paris XI, Orsay, France, 1994.

V. Fabian. Asymptotically efficient stochastic approximation; the RM case. *Annals of Statistics*, 1 (3):486–495, 1973.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proc. 25th Intl. Conf. on Machine Learning (ICML'08)*, pages 408–415. Omnipress, 2008.

Y. Le Cun, L. Bottou, G. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks, Tricks of the Trade*, Lecture Notes in Computer Science LNCS 1524. Springer Verlag, 1998.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *J. Machine Learning Research*, 5:361–397, 2004.

N. Murata and S.-I. Amari. Statistical analysis of learning dynamics. *Signal Processing*, 74(1): 3–28, 1999.

J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35:773–782, 1980.

N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *In Proc. of the 9th Intl. Conf. on Artificial Neural Networks*, pages 569–574, 1999.

N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-Newton method for online convex optimization. In *Proc. 11th Intl. Conf. on Artificial Intelligence and Statistics (AIstats)*, pages 433–440. Soc. for Artificial Intelligence and Statistics, 2007.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated subgradient solver for SVM. In *Proc. 24th Intl. Conf. on Machine Learning (ICML'07)*, pages 807–814. ACM, 2007.

S. Sonnenburg, V. Franc, E. Yom-Tov, and M. Sebag. Pascal large scale learning challenge. ICML'08 Workshop, 2008. `http://largescale.first.fraunhofer.de`.

Journal of Machine Learning Research 10 (2009) 1755-1758 Submitted 10/08; Revised 4/09; Published 7/09

Dlib-ml: A Machine Learning Toolkit

Davis E. King DAVISKING@USERS.SOURCEFORGE.NET
Northrop Grumman ES, ATR and Image Exploitation Group
Baltimore, Maryland, USA

Editor: Soeren Sonnenburg

Abstract

There are many excellent toolkits which provide support for developing machine learning software in Python, R, Matlab, and similar environments. Dlib-ml is an open source library, targeted at both engineers and research scientists, which aims to provide a similarly rich environment for developing machine learning software in the C++ language. Towards this end, dlib-ml contains an extensible linear algebra toolkit with built in BLAS support. It also houses implementations of algorithms for performing inference in Bayesian networks and kernel-based methods for classification, regression, clustering, anomaly detection, and feature ranking. To enable easy use of these tools, the entire library has been developed with contract programming, which provides complete and precise documentation as well as powerful debugging tools.

Keywords: kernel-methods, svm, rvm, kernel clustering, C++, Bayesian networks

1. Introduction

Dlib-ml is a cross platform open source software library written in the C++ programming language. Its design is heavily influenced by ideas from design by contract and component-based software engineering. This means it is first and foremost a collection of independent software components, each accompanied by extensive documentation and thorough debugging modes. Moreover, the library is intended to be useful in both research and real world commercial projects and has been carefully designed to make it easy to integrate into a user's C++ application.

There are a number of well known machine learning libraries. However, many of these libraries focus on providing a good environment for doing research using languages other than C++. Two examples of this kind of project are the Shogun (Sonnenburg et al., 2006) and Torch (Collobert and Bengio, 2001) toolkits which, while they are implemented in C++, are not focused on providing support for developing machine learning software in that language. Instead they are primarily intended to be used with languages like R, Python, Matlab, or Lua. Then there are toolkits such as Shark (Igel et al., 2008) and dlib-ml which are explicitly targeted at users who wish to develop software in C++. Given these considerations, dlib-ml attempts to help fill some of the gaps in tool support not already filled by libraries such as Shark. It is hoped that these efforts will prove useful for researchers and engineers who wish to develop machine learning software in this language.

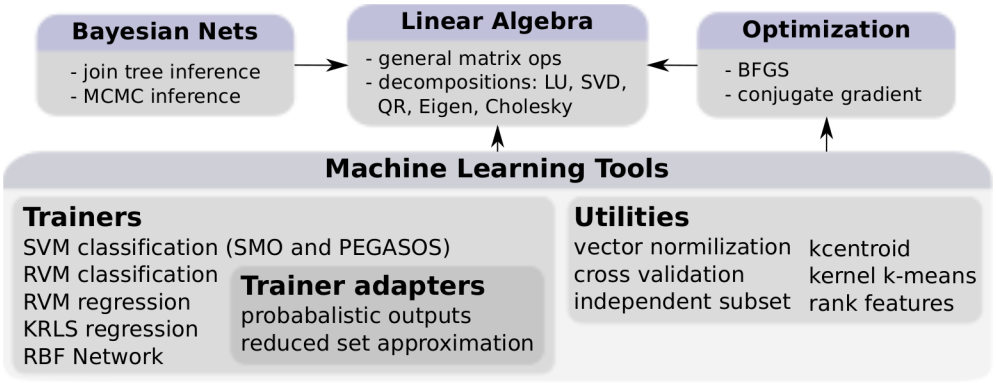


Figure 1: Elements of dlib-ml. Arrows show dependencies between components.

2. Elements of the Library

The library is composed of the four distinct components shown in Figure 1. The linear algebra component provides a set of core functionality while the other three implement various useful tools. This paper addresses the two main components, linear algebra and machine learning tools.

2.1 Linear Algebra

The design of the linear algebra component of the library is based on the template expression techniques popularized by Veldhuizen and Ponnambalam (1996) in the Blitz++ numerical software. This technique allows an author to write simple Matlab-like expressions that, when compiled, execute with speed comparable to hand-optimized C code. The dlib-ml implementation extends this original design in a number of ways. Most notably, the library can use the BLAS when available, meaning that the performance of code developed using dlib-ml can gain the speed of highly optimized libraries such as ATLAS or the Intel MKL while still using a very simple syntax. Consider the following example involving matrix multiplies, transposes, and scalar multiplications:

```
(1) result = 3*trans(A*B + trans(A)*2*B);
(2) result = 3*trans(B)*trans(A) + 6*trans(B)*A;
```

The result of expression (1) could be computed using only two calls to the matrix multiply routine in BLAS but first it is necessary to reorder the terms into form (2) to fit the form expected by the BLAS routines. Performing these transformations by hand is tedious and error prone. Dlib-ml automatically performs these transformations on all expressions and invokes the appropriate BLAS calls. This enables the user to write equations in the form most intuitive to them and leave these details of software optimization to the library. This is a feature not found in the supporting tools of other C++ machine learning libraries.

2.2 Machine Learning Tools

A major design goal of this portion of the library is to provide a highly modular and simple architecture for dealing with kernel algorithms. In particular, each algorithm is parameterized to allow a user to supply either one of the predefined dlib-ml kernels, or a new user defined kernel. Moreover, the implementations of the algorithms are totally separated from the data on which they operate.

This makes the dlib-ml implementation generic enough to operate on any kind of data, be it column vectors, images, or some other form of structured data. All that is necessary is an appropriate kernel.

This is a feature unique to dlib-ml. Many libraries allow arbitrary precomputed kernels and some even allow user defined kernels but have interfaces which restrict them to operating on column vectors. However, none allow the flexibility to operate *directly* on arbitrary objects, making it much easier to apply custom kernels in the case where the kernels operate on objects other than fixed length vectors.

The library provides implementations of popular algorithms such as RBF networks and support vector machines for classification. It also includes algorithms not present in other major ML toolkits such as relevance vector machines for classification and regression (Tipping and Faul, 2003). All of these algorithms are implemented as generic trainer objects with a standard interface. This design allows trainer objects to be used by a number of generic meta-algorithms that do tasks such as performing cross validation, reducing the number of output support vectors (Suttorp and Igel, 2007), or fitting a sigmoid to the output decision function to make decisions interpretable in probabilistic terms (Platt, 1999). This generic trainer interface, along with the contract programming approach, makes the library easily extensible by other developers.

Another good example of a generic kernel algorithm provided by the library is the kernel RLS technique introduced by Engel et al. (2004). It is a kernelized version of the famous recursive least squares filter, and functions as an excellent online regression method. With it, Engel introduced a simple but very effective technique for producing sparse outputs from kernel learning algorithms.

Engel's sparsification technique is also used by one of dlib-ml's most versatile tools, the kcentroid object. It is a general utility for representing a weighted sum of sample points in a kernel induced feature space. It can be used to easily kernelize any algorithm that requires only the ability to perform vector addition, subtraction, scalar multiplication, and inner products.

The kcentroid object enables the library to provide a number of useful kernel-based machine learning algorithms. The most straightforward of which is online anomaly detection, which simply marks data samples as novel if their distance from the centroid of a previously observed body of data is large (e.g., 3 standard deviations from the mean distance). A similarly simple but still powerful application is in feature ranking, where features are considered good if their inclusion results in a large distance between the centroids of different classes of data.

Another straightforward application of this technique is in kernelized cluster analysis. Using the kcentroid it is easy to create sparse kernel clustering algorithms. To demonstrate this, the library comes with a sparse kernel k-means algorithm.

Finally, dlib-ml contains two SVM solvers. One is essentially a reimplementation of LIBSVM (Chang and Lin, 2001) but with the generic parameterized kernel approach used in the rest of the library. This solver has roughly the same CPU and memory utilization characteristics as LIBSVM. The other SVM solver is a kernelized version of the Pegasos algorithm introduced by Shalev-Shwartz et al. (2007). It is built using the kcentroid and thus produces sparse outputs.

3. Availability and Requirements

The library is released under the Boost Software License, allowing it to be incorporated into both open-source and commercial software. It requires no additional libraries, does not need to be configured or installed, and is frequently tested on MS Windows, Linux and MacOS X but should work with any ISO C++ compliant compiler.

Note that dlib-ml is a subset of a larger project named dlib hosted at http://dclib.sourceforge.net. Dlib is a general purpose software development library containing a graphical application for creating Bayesian networks as well as tools for handling threads, network I/O, and numerous other tasks. Dlib-ml is available from the dlib project's download page on SourceForge.

References

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Ronan Collobert and Samy Bengio. Svmtorch: support vector machines for large-scale regression problems. *J. Mach. Learn. Res.*, 1:143–160, 2001. ISSN 1533-7928.

Yaakov Engel, Shie Mannor, and Ron Meir. Kernel recursive least squares. *IEEE Transactions on Signal Processing*, 52:2275–2285, 2004.

Christian Igel, Tobias Glasmachers, and Verena Heidrich-Meisner. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML '07*, pages 807–814, New York, NY, USA, 2007. ACM.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *J. Mach. Learn. Res.*, 7:1531–1565, 2006. ISSN 1533-7928.

Thorsten Suttorp and Christian Igel. Resilient approximation of kernel classifiers. volume 4668 of *Lecture Notes in Computer Science*, pages 139–148. Springer, 2007.

Michael E. Tipping and Anita C. Faul. Fast marginal likelihood maximisation for sparse Bayesian models. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, pages 3–6, 2003.

Todd Veldhuizen and Kumaraswamy Ponnambalam. Linear algebra with C++ template metaprograms. *Dr. Dobb's Journal of Software Tools*, 21(8):38–44, 1996.

Journal of Machine Learning Research 10 (2009) 1759-1799 Submitted 10/07; Revised 3/09; Published 8/09

Settable Systems: An Extension of Pearl's Causal Model with Optimization, Equilibrium, and Learning

Halbert White HWHITE@UCSD.EDU
Department of Economics
University of California, San Diego
9500 Gilman Drive
La Jolla, CA 92093, USA

Karim Chalak CHALAK@BC.EDU
Department of Economics
Boston College
140 Commonwealth Avenue
Chestnut Hill, MA 02467, USA

Editor: Michael Jordan

Abstract

Judea Pearl's Causal Model is a rich framework that provides deep insight into the nature of causal relations. As yet, however, the Pearl Causal Model (PCM) has had a lesser impact on economics or econometrics than on other disciplines. This may be due in part to the fact that the PCM is not as well suited to analyzing structures that exhibit features of central interest to economists and econometricians: optimization, equilibrium, and learning. We offer the settable systems framework as an extension of the PCM that permits causal discourse in systems embodying optimization, equilibrium, and learning. Because these are common features of physical, natural, or social systems, our framework may prove generally useful for machine learning. Important features distinguishing the settable system framework from the PCM are its countable dimensionality and the use of partitioning and partition-specific response functions to accommodate the behavior of optimizing and interacting agents and to eliminate the requirement of a unique fixed point for the system. Refinements of the PCM include the settable systems treatment of attributes, the causal role of exogenous variables, and the dual role of variables as causes and responses. A series of closely related machine learning examples and examples from game theory and machine learning with feedback demonstrates some limitations of the PCM and motivates the distinguishing features of settable systems.

Keywords: causal models, game theory, machine learning, recursive estimation, simultaneous equations

1. Introduction

Judea Pearl's work on causality, especially as embodied in his landmark book *Causality* (Pearl, 2000), represents a rich framework in which to understand, analyze, and explain causal relations. This framework has been adopted and applied in a broad array of disciplines, but so far it has had a lesser impact in economics. This may be due in part to the fact that the Pearl causal model (PCM) is not as explicit about or well suited to analyzing structures that exhibit features of central interest to economists and econometricians: optimization, equilibrium, and learning. Here, we offer the

settable systems framework as an extension of the PCM that permits causal discourse in systems embodying these features.

Because optimization, equilibrium, and learning are features not only of economic systems, but also of physical, natural, or social systems more generally, our extended framework may prove useful elsewhere, especially in areas where empirical analysis, whether observational or experimental, has a central role to play. In particular, settable systems offer a number of advantages relative to the PCM for machine learning. To show this, we provide a detailed examination of the features and limitations of the PCM relevant to machine learning. This examination provides key insight into the PCM and helps to motivate features of the settable systems framework we propose.

Roughly speaking, a settable system is a mathematical framework describing an environment in which multiple agents interact under uncertainty. In particular, the settable systems framework is explicit about the principles underlying how agents make decisions, the equilibria (if any) resulting from agents' decisions, and learning from repeated interactions. Because it is explicit about agents' decision making, the settable systems framework extends the PCM by providing a decision-theoretic foundation for causal analysis (see, e.g., Heckerman and Shachter, 1995) in the spirit of influence diagrams (Howard and Matheson, 1984). However, unlike influence diagrams, the settable systems framework preserves the spirit of the PCM and its appealing features for empirical analysis, including its use of response functions and the causal notions that these support.

As Koller and Milch (2003, pp. 189-190) note in motivating their study of multi-agent influence diagrams (MAIDs), "influence diagrams [...] have been investigated almost entirely in a single-agent setting." The settable systems framework also permits the study of multiple agent interactions. Nevertheless, a number of settable systems features distinguishes them from MAIDs, as we discuss in Section 6.4. Among other things, settable systems permit causal discourse in systems with multi-agent interactions.

Some features of settable systems are entirely unavailable in the PCM. These include (1) accommodating an infinite number of agents; and (2) the absence of a unique fixed point requirement. Other features of settable systems rigorously formalize and refine or extend related PCM features, thereby permitting a more explicit causal discourse. These features include (3) the notion of attributes, (4) definitions of interventions and direct effects, (5) the dual role of variables as causes and responses, and (6) the causal role of exogenous variables.

For instance, for a given system, the PCM's common treatment of attributes and background variables rules out a causal role for background variables. Specifically, this rules out structurally exogenous causes, whether observed or unobserved. This also limits the role of attributes in characterizing systems of interest. Because the status of a variable in the PCM is relative to the analysis and is entirely up to the researcher, a background variable may be treated as an endogenous variable in an alternative system if deemed sensible by the researcher, thereby permitting it to have a causal role. Nevertheless, the PCM is silent about how to distinguish between attributes, background variables, and endogenous variables. In contrast, in settable systems one or more governing principles, such as optimization or equilibrium, provide a formal and explicit way to distinguish between structurally exogenous and endogenous variables, permitting explicitly causal roles not only for endogenous but also for exogenous variables. Attributes are unambiguously defined as constants (numbers, sets, functions) associated with the system units that define fundamental aspects of the decision problem represented by the settable system.

The Rubin treatment effect approach to causal inference (e.g., as formalized by Holland, 1986) also relates to settable systems. We leave a careful study of the relations between these two ap-

proaches to other work in order to keep a sharp and manageable focus for this paper. Thus, our goal here is to compare and contrast our approach with the PCM, which, along with structural equation systems in econometrics, comprise the frameworks that primarily motivate settable systems. Nevertheless, some brief discussion of the relation of settable systems to Rubin's treatment effect approach is clearly warranted. In our view, the main feature that distinguishes settable systems (and the PCM) from the Rubin model is the explicit representation of the full causal structure. This has significant implications for the selection of covariates and for providing primitive conditions that deliver unconfoundedness conditions as consequences in settable systems, rather than introducing these as maintained assumptions in the Rubin model. Explicit representation of the full causal structure also has important implications for the analysis of "simultaneous" systems and mutually causal relations, which are typically suppressed in the Rubin approach. Finally, the allowance for a countable number of system units, the partitioning device of settable systems, and settable systems' more thorough exploitation of attributes also represent useful differences with Rubin's model.

The plan of this paper is as follows. In Section 2, we give a succinct statement of the elements of the PCM and of a generalization due to Halpern (2000) relevant for motivating and developing our settable systems extension.

Section 3 contains a series of closely related machine learning examples in which we examine the features and limitations of the PCM. These in turn help motivate features of the settable systems framework. Our examples involve least squares-based machine learning algorithms for simple artificial neural networks useful for making predictions. We consider learning algorithms with and without weight clamping and network structures with and without hidden units. Because learning is based on principles of optimization (least squares), our discussion relates to decision problems generally.

Our examples in Section 3 show that although the PCM applies to key aspects of machine learning, it also fails to apply to important classes of problems. One source of these limitations is the PCM's unique fixed point requirement. Although Halpern's (2000) generalization does not impose this requirement, it has other limitations. We contrast these with settable systems, where there is no fixed point requirement, but where fixed points may help determine system outcomes. The feature of settable systems delivering this flexibility is partitioning, an analog of the submodel and do operator devices of the PCM.

The examples of Section 3 do not involve randomness. We introduce randomness in Section 4, using our machine learning examples to discuss heuristic aspects of stochastic settable systems. We compare and contrast these with aspects of Pearl's probabilistic causal model. An interesting feature of stochastic settable systems is that attributes can determine the governing probability measure. In contrast, attributes are random variables in the PCM. Straightforward notions of counterfactuals, interventions, direct causes, direct effects, and total effects emerge naturally from stochastic settable systems.

Section 5 integrates the features of settable systems motivated by our examples to provide a rigorous formal definition of stochastic settable systems.

In Section 6 we use a series of examples from game theory to show how settable systems apply to groups of interacting and strategically competitive decision-making agents. Game theoretic structures have broad empirical relevance; they also present interesting opportunities for distributed and emergent computation of important quantities, such as prices. The decision-making agents may be consumers, firms, or government entities; they may also be biological systems or artificial intelligences, as in automated trading systems. Our demonstrations thus provide foundations for causal

analysis of systems where optimization and equilibrium mechanisms both operate to determine system outcomes. We relate our results to multi-agent influence diagrams (Koller and Milch, 2003) in Section 6.4.

In Section 7 we close the loop by considering examples from a general class of machine learning algorithms with feedback introduced by Kushner and Clark (1978) and extended by Chen and White (1998). These systems contain not only learning methods involving possibly hidden states, such as the Kalman filter (Kalman, 1960) and recurrent neural networks (e.g., Elman, 1990; Jordan, 1992; Kuan, Hornik, and White, 1994), but also systems of groups of strategically interacting and learning decision makers, as shown by Chen and White (1998). These systems exhibit optimization, equilibrium, and learning and map directly to settable systems, providing foundations for causal analysis in such systems.

Section 8 contains a summary and a discussion of research relying on the foundations provided here as well as discussion of directions for future work. An Appendix contains supplementary material; specifically, we give a formal definition of nonstochastic settable systems.

In a recent review of Pearl's book for economists and econometricians, Neuberg (2003) expresses a variety of reservations and concerns. Nevertheless, Neuberg (2003, p. 685) recommends that "econometricians should read *Causality* and start contributing to the cross-disciplinary discussion of the subject that Pearl has begun. Hopefully mutual enlightenment will be the effect of our reading and talking about the book among ourselves and with the Bayesian causal network thinkers." By examining aspects of what can and cannot be accommodated within Pearl's framework, and by proposing settable systems as an extension of this framework designed to accommodate features of central interest to economists, namely optimization, equilibrium, and learning, we offer this paper as part of this dialogue.

2. Pearl's Causal Model

Pearl's definition of a *causal model* (Pearl, 2000, Def. 7.1.1, p. 203) provides a formal statement of the elements essential to causal reasoning. According to this definition, a causal model is a triple $M := (u, v, f)$, where $u := \{u_1, ..., u_m\}$ is a collection of "background" variables determined outside the model, $v := \{v_1, ..., v_n\}$ is a collection of "endogenous" variables determined within the model, and $f := \{f_1, ..., f_n\}$ is a collection of "structural" functions that specify how each endogenous variable is determined by the other variables of the model, so that $v_i = f_i(v_{(i)}, u), \quad i = 1, ..., n$. Here $v_{(i)}$ denotes the vector containing every element of v except v_i. The integers m and n are finite. We refer to the elements of u and v as system "units."

Finally, the definition requires that f yields a unique fixed point for each u, so that there exists a unique collection $g := \{g_1, ..., g_n\}$ such that for each u,

$$v_i = g_i(u) = f_i(g_{(i)}(u), u), \quad i = 1, ..., n.$$

The unique fixed point requirement is a crucial aspect of the PCM, as this ensures existence of the *potential response function* (Pearl, 2000, Def. 7.1.4). This provides the foundation for discourse about causal relations between endogenous variables; this discourse is not possible in the PCM otherwise. A variant of the PCM analyzed by Halpern (2000) does not require a fixed point, but if any exist, there may be multiple collections of functions g yielding a fixed point. We refer to such a model as a Generalized Pearl Causal Model (GPCM). We note that GPCMs do not possess an analog of the potential response function, due to the lack of a unique fixed point.

In presenting the elements of the PCM, we have adapted Pearl's original notation somewhat to facilitate the discussion to follow, but all essential elements of the definition are present and complete.

3. Machine Learning, the PCM, and Settable Systems

We now consider how machine learning can be viewed in the context of the PCM. We consider machine learning examples that fundamentally involve optimization, a feature of a broad range of physical, natural, and social systems.[1] Specifically, optimization lies at the heart of most decision problems, as these problems typically involve deciding which of a range of possible options delivers the best expected outcome given the available information. When machine learning is based on optimization, it represents a prototypical decision problem. As we show, certain important aspects of machine learning map directly to the PCM. This permits us to investigate which causal questions are meaningful for machine learning within the PCM, and it motivates the modifications and refinements that lead to settable systems and the more extensive causal discourse possible there.

3.1 A Least-Squares Learning Example

Our first example considers predicting a random variable Y using a single random predictor X and an artificial neural network. In particular, we study the causal consequences for the optimal network weights of interventions to certain parameters of the joint distribution of the randomly generated X and Y.

More specifically, the output of an artificial neural network having a simple linear architecture is given by

$$f(X;\alpha,\beta) = \alpha + \beta X.$$

We suppose that Y and X are randomly generated according to a joint distribution F_γ indexed by a vector of parameters γ belonging to the parameter space Γ. We thus view γ as a variable whose values may range over Γ. For clarity, we suppose that γ is not influenced by our prediction (e.g., a weather forecast or an economic growth forecast).

We evaluate network performance in terms of expected squared prediction error loss,

$$\begin{aligned} L(\alpha,\beta,\gamma) &:= E_\gamma([Y - f(X;\alpha,\beta)]^2) \\ &= \int [y - f(x;\alpha,\beta)]^2 dF_\gamma(x,y), \end{aligned}$$

where $E_\gamma(\cdot)$ denotes expectation taken with respect to the distribution F_γ. Our goal is to obtain the best possible predictions according to this criterion. Accordingly, we seek loss-minimizing network weights, which solve the optimization problem

$$\min_{\alpha,\beta} L(\alpha,\beta,\gamma).$$

This makes it explicit that the governing principle in this example is optimization.

Under mild conditions, least squares-based machine learning algorithms converge to the optimal weights as the size of the training data set grows. For clarity, we work for now with the optimal network weights.

1. The great mathematician Leonhard Euler once wrote, "nothing at all takes place in the Universe in which some rule of maximum or minimum does not appear" (as quoted in Marsden and Tromba, 2003).

For our linear network, the first order conditions necessary for an optimum are

$$\begin{aligned}(\partial/\partial\alpha)L(\alpha,\beta,\gamma) &= -2E_\gamma([Y-\alpha-\beta X]) = 0,\\ (\partial/\partial\beta)L(\alpha,\beta,\gamma) &= -2E_\gamma(X[Y-\alpha-\beta X]) = 0.\end{aligned}$$

Letting $\mu_X := E_\gamma(X)$, $\mu_Y := E_\gamma(Y)$, $\mu_{XX} := E_\gamma(X^2)$, and $\mu_{XY} := E_\gamma(XY)$, we can conveniently parameterize F_γ in terms of the moments $\gamma := (\mu_X, \mu_Y, \mu_{XX}, \mu_{XY})$. (This parameterization need not uniquely determine F_γ; that is, there may be multiple distributions F_γ for a given γ. Nevertheless, this γ is the only aspect of the distribution that matters here.) We can then express the first order conditions equivalently as

$$\begin{aligned}\mu_Y - \alpha - \beta\mu_X &= 0,\\ \mu_{XY} - \alpha\mu_X - \beta\mu_{XX} &= 0.\end{aligned}$$

Now consider how this system fits into Pearl's causal model. Pearl's model requires a system of equations in which the left-hand side variables are structurally determined by the right-hand side variables. The first order conditions are not in this form, but, provided $\mu_{XX} - \mu_X^2 > 0$, they can be transformed to this form by solving jointly for α and β :

$$\begin{aligned}\alpha^* &= \mu_Y - [\mu_{XX} - \mu_X^2]^{-1}(\mu_{XY} - \mu_X\mu_Y)\mu_X,\\ \beta^* &= [\mu_{XX} - \mu_X^2]^{-1}(\mu_{XY} - \mu_X\mu_Y).\end{aligned} \tag{1}$$

We write (α^*, β^*) to distinguish optimized values from generic values (α, β).

This representation demonstrates that the PCM applies directly to this machine learning problem. The equations in (1) form a system in which the background (or "structurally exogenous") variables $u := (u_1, u_2, u_3, u_4) = (\mu_X, \mu_Y, \mu_{XX}, \mu_{XY}) =: \gamma$ determine the endogenous variables $v := (v_1, v_2) = (\alpha^*, \beta^*)$. The structural functions (f_1, f_2) are defined by

$$\begin{aligned}f_1(u) &= u_2 - [u_3 - u_1^2]^{-1}(u_4 - u_1u_2)u_1,\\ f_2(u) &= [u_3 - u_1^2]^{-1}(u_4 - u_1u_2).\end{aligned}$$

We observe that by the conventions of the PCM, the background variables u do not have formal status as causes, as we further discuss below.

In discussing the PCM, Pearl (2000, p. 203) notes that the background variables are often unobservable, but this is not a formal requirement of the PCM. In our example, we may view the γ variables as either observable or unobservable, depending on the context. For example, suppose we are given a linear least-squares learning machine as a black box: we know that it is a learning machine, but we don't know of what kind. To attempt to determine what is inside the black box, we can conduct computer experiments in which we set γ to various known values and observe the resulting values of (α^*, β^*). In this case, γ is observable.

Alternatively, we may have a least-squares learning machine that we apply to a variety of data sets obeying the distribution F_γ for differing unknown values of γ. In each case, γ is unobservable, but we can generate as much data as we want from F_γ.

Intermediate cases are also possible, in which some elements of γ are known and others are not. For example, in the multiple data set example, we could have knowledge of a subvector of γ, for example, we might know μ_X.

3.2 Learning with Clamping

Next, we study the effects on one of the optimal network weights of interventions to the other weight. For this, we consider the optimal network weights that arise when one or the other of the network weights is clamped, that is, set to an arbitrary fixed value. Specifically, consider

$$\min_{\alpha} L(\alpha,\beta,\gamma) \text{ and } \min_{\beta} L(\alpha,\beta,\gamma).$$

Clamping is useful in "nested" or multi-stage optimization, as

$$\begin{aligned} \min_{\alpha,\beta} L(\alpha,\beta,\gamma) &= \min_{\beta}[\min_{\alpha} L(\alpha,\beta,\gamma)] \text{ and} \\ \min_{\alpha,\beta} L(\alpha,\beta,\gamma) &= \min_{\alpha}[\min_{\beta} L(\alpha,\beta,\gamma)]. \end{aligned}$$

See, for example, Sergeyev and Grishagin (2001). Clamping is a central feature of a variety of powerful machine learning algorithms, for example, the restricted Boltzmann machine (e.g., Ackley et al., 1985; Hinton and Sejnowski, 1986; Hinton et al., 2006; Hinton and Salakhutdinov, 2006). Learning in stages is particularly useful in cases involving complex optimizations, as in the EM algorithm (Dempster, Laird, and Rubin, 1977).

The first order condition necessary for the β-clamped optimum $\min_\alpha L(\alpha,\beta,\gamma)$ is

$$(\partial/\partial\alpha)L(\alpha,\beta,\gamma) = -2E_\gamma([Y-\alpha-\beta X]) = 0.$$

Equivalently, $\mu_Y - \alpha - \beta\mu_X = 0$. Solving for the optimal α weight gives

$$\tilde{\alpha}^* = \mu_Y - \beta\mu_X. \tag{2}$$

We use the tilde notation to distinguish between the optimal weights with clamping and the jointly optimal weights obtained above.

Similarly, the first order condition necessary for the α-clamped optimum $\min_\beta L(\alpha,\beta,\gamma)$ is

$$(\partial/\partial\beta)L(\alpha,\beta,\gamma) = -2E_\gamma(X[Y-\alpha-\beta X]) = 0.$$

Equivalently, $\mu_{XY} - \alpha\mu_X - \beta\mu_{XX} = 0$. Given $\mu_{XX} > 0$, the optimal weight with clamping is

$$\tilde{\beta}^* = \mu_{XX}^{-1}(\mu_{XY} - \alpha\mu_X). \tag{3}$$

Writing Equations (2) and (3) as a system, we have

$$\tilde{\alpha}^* = \mu_Y - \beta\mu_X \qquad \tilde{\beta}^* = \mu_{XX}^{-1}(\mu_{XY} - \alpha\mu_X). \tag{4}$$

This resembles a structural system in the form of the PCM, except that here $\tilde{\alpha}^*$ and $\tilde{\beta}^*$ appear on the left, instead of α and β. This difference is significant; we address this shortly.

Nevertheless, suppose for the moment that we ignore this difference and modify the system above to conform to the PCM by replacing $\tilde{\alpha}^*$ and $\tilde{\beta}^*$ with α and β :

$$\alpha = \mu_Y - \beta\mu_X \qquad \beta = \mu_{XX}^{-1}(\mu_{XY} - \alpha\mu_X). \tag{5}$$

We take $u = \gamma$ as above, but in keeping with our conforming modification, we now take $(v_1, v_2) = (\alpha, \beta)$. The structural functions become

$$\tilde{f}_1(u, v_2) = u_2 - v_2 u_1 \qquad \tilde{f}_2(u, v_1) = u_3^{-1}(u_4 - v_1 u_1).$$

This system falls into the PCM, with consequent causal status for v, provided there is a unique fixed point for each u.

Unfortunately, this fixed point requirement fails here. As is apparent from the equations in (5), the only necessary restriction on u is that $u_3 = \mu_{XX} > 0$. This is the requirement that X is not equal to 0 with probability one. Nevertheless, it is readily verified that even with this restriction, the fixed point requirement fails for all u such that

$$u_3 - u_1^2 = \mu_{XX} - \mu_X^2 = 0.$$

This is the condition that $X = \mu_X$ with probability one, and μ_X can take any value, not just zero. When this condition holds, there is an uncountable infinity of fixed point solutions to the equations in (5). Stated another way, the solution to the system is set-valued in this circumstance.

Because of the lack of a fixed point, the PCM does not apply and therefore cannot provide causal meaning for such a system. The inability of the PCM to apply to this simple example of machine learning with clamping is an unfortunate limitation. Because Halpern's (2000) GPCM does not require a unique fixed point, it does apply here. Nevertheless, the lack of the potential response function in the GPCM prevents the desired causal discourse.

3.3 Settable Systems and Learning with Clamping

We now consider how these issues can be addressed. Our intent is to encompass this example while preserving the spirit of the PCM. This motivates and helps illustrate various features of our settable systems framework.

3.3.1 SETTABLE VARIABLES

We begin by taking seriously the difference in roles between (α, β) and $(\tilde{\alpha}^*, \tilde{\beta}^*)$ appearing in the equations in (4). In the simplest sense, the difference is that $(\tilde{\alpha}^*, \tilde{\beta}^*)$ and (α, β) appear on different sides of the equal signs: (α, β) appears on the right and $(\tilde{\alpha}^*, \tilde{\beta}^*)$ on the left. In the PCM, this difference is fundamentally significant, in that causal relations are asymmetric, with structurally determined (endogenous) variables on the left and all other variables on the right. In settable systems, we formalize these dual roles by defining *settable variables* as mappings $\mathcal{X}$ with a dual aspect:

$$\begin{aligned} \mathcal{X}_1(0) &:= \tilde{\alpha}^*, \quad \mathcal{X}_1(1) := \alpha, \\ \mathcal{X}_2(0) &:= \tilde{\beta}^*, \quad \mathcal{X}_2(1) := \beta. \end{aligned} \tag{6}$$

We call the $0-1$ argument of the settable variables $\mathcal{X}$ the "role indicator." When this is 0, the value of the variable is that determined by its structural equation. We call these values *responses*. In contrast, when the role indicator is 1, the value is not determined by its structural equation, but is instead set to one of its admissible values. We call these values *settings*. We require that a setting has *more than one* admissible value. That is, settings are variable.

Formally distinguishing between responses and settings makes explicit the dual roles played by variables in a causal system, entirely in the spirit of the PCM. Settable variables represent a formal

implementation, alternative to that of the "do operator" in the PCM, of the "wiping out" operation first proposed by Strotz and Wold (1960) and later used by Fisher (1970).

Once we make explicit the dual roles of the system variables, several benefits become apparent. First, the equal sign no longer has to serve in an asymmetric manner. This makes possible *implicit* representations of causal relations in settable systems that are either not possible in the PCM, because the required closed-form expressions do not exist; or that are possible in the PCM only under restrictions permitting application of the implicit function theorem. Such implicit representations are often natural for responses satisfying first order conditions arising from optimization. To illustrate, consider how explicit representation of the dual roles of system variables modifies the learning with clamping system. The first order condition necessary for the β-clamped optimum $\min_\alpha L(\alpha,\beta,\gamma)$ is now

$$\mu_Y - \tilde{\alpha}^* - \beta\mu_X = 0.$$

That for the α-clamped optimum $\min_\beta L(\alpha,\beta,\gamma)$ is now

$$\mu_{XY} - \alpha\mu_X - \tilde{\beta}^*\mu_{XX} = 0.$$

The structural system thus has the implicit representation

$$\mu_Y - \tilde{\alpha}^* - \beta\mu_X = 0, \tag{7}$$

$$\mu_{XY} - \alpha\mu_X - \tilde{\beta}^*\mu_{XX} = 0. \tag{8}$$

3.3.2 SETTABLE SYSTEMS AND THE ROLE OF FIXED POINTS

A second benefit of making explicit the dual roles of the system variables is that unique fixed points do not have a crucial role to play in settable systems. This enables us to dispense with the unique fixed point requirement prohibiting the PCM from encompassing our learning with clamping example. This is not to say that fixed points have no role to play. Instead, that role is removed from the structural representation of the system and, to the extent relevant, operates according to the governing principle, for example, optimization or equilibrium. We discuss this further below.

To illustrate, consider the learning with clamping system above where the dual roles of the system variables are made explicit. Now there is no necessity of finding a fixed point for Equations (7) and (8). Each equation stands on its own, representing its associated clamped optimum.

The simplest case is that for $\tilde{\alpha}^*$. For every μ_X, μ_Y, and β, there is a unique solution,

$$\tilde{\alpha}^* = \mu_Y - \beta\mu_X =: \tilde{r}_1(\beta,\gamma).$$

We call $\tilde{r}_1$ the *response function* for $\mathcal{X}_1$.

Next consider $\tilde{\beta}^*$. Provided $\mu_{XX} > 0$, Equation (8) determines a unique value for $\tilde{\beta}^*$,

$$\tilde{\beta}^* = \mu_{XX}^{-1}(\mu_{XY} - \alpha\mu_X).$$

But what happens when $\mu_{XX} = 0$? This further implies $\mu_X = \mu_{XY} = 0$. Consequently, *any* value will do for $\tilde{\beta}^*$, as any value of $\tilde{\beta}^*$ delivers the best possible prediction. To arrive at a unique value for $\tilde{\beta}^*$, we can apply criteria supplemental to predictive optimality. For example, we may choose a value that has the simplest representation. This reduces the viable choices to $\tilde{\beta}^* \in \{0,1\}$, as either of these requires only one bit to represent. Finally, by selecting $\tilde{\beta}^* \in \{0\}$, so that we set $\tilde{\beta}^* = 0$ when

$\mu_{XX} = 0$, we achieve a prediction, $f(X;\alpha,\tilde{\beta}^*) = \alpha$, that requires the fewest operations to compute. Formally, this gives

$$\tilde{\beta}^* = 1_{\{\mu_{XX}>0\}}\mu_{XX}^{-1}(\mu_{XY} - \alpha\mu_X) =: \tilde{r}_2(\alpha,\gamma),$$

where $1_{\{\mu_{XX}>0\}}$ is the indicator function taking the value one when $\mu_{XX} > 0$, and zero otherwise. We call $\tilde{r}_2$ the response function for $\mathcal{X}_2$.

This example demonstrates that even when structural equations conforming to the PCM (i.e., Equation 5) do not have a fixed point, we can find unique response functions for each settable variable of the analogous settable system. We do this by applying the governing principle for the system (e.g., optimization), supplemented when necessary by further appropriate principles (e.g., parsimony of memory and computation).

Applying the settable variable representation in the equations in (6), we obtain a settable variables representation for our learning with clamping example:

$$\mathcal{X}_1(0) = \tilde{r}_1(\mathcal{X}_2(1),\gamma), \qquad \mathcal{X}_2(0) = \tilde{r}_2(\mathcal{X}_1(1),\gamma).$$

So far, the variable γ has not been given status as a settable variable. Although it does not have a dual aspect, it can be set to any of several admissible values (those in Γ), so it does have the aspect of a setting. Accordingly, we can define $\mathcal{X}_0(1) := \gamma$. To ensure that $\mathcal{X}_0$ is a well-defined settable variable, we must also specify a value for $\mathcal{X}_0(0)$. By convention, we simply put $\mathcal{X}_0(0) := \mathcal{X}_0(1)$. We call $\mathcal{X}_0$ *fundamental* settable variables. As these are determined outside the system, they are structurally exogenous.

We can now give an explicit settable system representation for our present example, that is, a representation solely in terms of settable variables:

$$\mathcal{X}_1(0) = \tilde{r}_1(\mathcal{X}_2(1),\mathcal{X}_0(1)) \qquad \mathcal{X}_2(0) = \tilde{r}_2(\mathcal{X}_1(1),\mathcal{X}_0(1)).$$

3.4 Causes and Effects: Settable Systems and the PCM

This section introduces causal notions appropriate to settable systems.

3.4.1 DIRECT CAUSALITY

We begin by considering our learning with clamping example, where

$$\tilde{\alpha}^* = \tilde{r}_1(\beta,\gamma), \quad \tilde{\beta}^* = \tilde{r}_2(\alpha,\gamma).$$

In particular, consider the equation $\tilde{\beta}^* = \tilde{r}_2(\alpha,\gamma)$. In settable systems, settings are variable, that is, they can take any of a range of admissible values. We view this as sufficient to endow them with potential causal status. Thus, we call α and γ *potential causes* of $\tilde{\beta}^*$.

We say that a given element of (α,γ) *does not directly cause* $\tilde{\beta}^*$ if $\tilde{r}_2(\alpha,\gamma)$ defines a function constant in the given element for all admissible values of the other elements of (α,γ). Otherwise, that element is a *direct cause* of $\tilde{\beta}^*$. According to this definition, μ_Y does not directly cause $\tilde{\beta}^*$, whereas $\mu_X, \mu_{XX}, \mu_{XY}$, and α are direct causes of $\tilde{\beta}^*$.

3.4.2 INTERVENTIONS AND DIRECT EFFECTS IN SETTABLE SYSTEMS

In settable systems, an *intervention to a settable variable* is a pair of distinct admissible setting values. In our clamped learning example, let α_1 and α_2 be different admissible values for α.

Then $\alpha_1 \to \alpha_2 := (\alpha_1, \alpha_2)$ is an intervention to α, or, more formally, to $\mathcal{X}_1$. Similarly, $(\alpha_1, \gamma_1) \to (\alpha_2, \gamma_2) := ((\alpha_1, \gamma_1), (\alpha_2, \gamma_2))$ is an intervention to (α, γ) (i.e., to $(\mathcal{X}_1, \mathcal{X}_0)$). The *direct effect* on a given settable variable of a specified intervention is the response difference arising from the intervention. In our clamped learning example, the direct effect on $\mathcal{X}_2$ of the intervention $\alpha_1 \to \alpha_2$ is

$$\begin{aligned} \Delta\tilde{r}_2(\alpha_1, \alpha_2; \gamma) &:= \tilde{r}_2(\alpha_2, \gamma) - \tilde{r}_2(\alpha_1, \gamma) \\ &= 1_{\{\mu_{XX} > 0\}} \mu_{XX}^{-1} (\alpha_1 - \alpha_2) \mu_X. \end{aligned}$$

We emphasize that interventions are always well defined, as settings necessarily have more than one admissible value. Indeed, a key reason that we require settings to be variable is precisely to ensure that interventions to settable variables are always meaningful.

PCM notions related to the settable systems notion of intervention are the do operator and the "effect of action" defined in definition 7.1.3 of Pearl (2000); these specify a submodel associated with a given realization x for a given subset of the endogenous variables v.

3.4.3 EXOGENOUS AND ENDOGENOUS CAUSES

The notion of causality just defined contrasts in an interesting way with that formally given in the PCM. We have just seen that γ can serve in the settable system as a direct cause of $\tilde{\beta}^*$. Above, we saw that γ corresponds to background variables u in the PCM. In the PCM, the formal concept of *submodel* and the *do operator* necessary to define causal relations are meaningful only for endogenous variables v. None of these concepts are defined for u; that is, u is not subject to counterfactual variation in the PCM.[2] Consequently, u does not have formal causal status in the PCM as defined in Pearl (2000, Chap. 7).

In the PCM, u thus has four explicit distinguishing features: it is (i) a vector of variables that (ii) are determined outside the system, (iii) determine the endogenous variables, and (iv) are not subject to counterfactual variation. An optional but common feature of u is: (v) it is unobservable. As a result, background variables cannot act as causes in the PCM; in particular, for a given system, the PCM formally rules out structurally exogenous unobserved causes.

In settable systems, we drop requirement (iv) for structurally exogenous variables. Thus, we allow for observed structurally exogenous causes such as a treatment of interest in a controlled experiment, which is typically directly set (and observed) by the researcher. We also allow for unobservable structurally exogenous causes, ensuring a causal framework that is not relative to the capabilities of the observer, as is appropriate to the macroscopic, non-quantum mechanical systems that are the strict focus of our attention here. Unobserved common causes are particularly relevant for the analysis of confounding, that is, the existence of hidden causal relations that may prevent the identification of causal effects of interest (see Pearl, 2000, Chap. 3.3-3.5). Also, unobserved structurally exogenous causes are central to errors-in-variables models where a structurally exogenous cause of interest cannot be observed. Instead, one observes an version of this cause contaminated by measurement error. These models are the subject of a vast literature in statistics and econometrics (see, e.g., van Huffel and Lemmerling, 2002, and the references there).

Dropping (iv) in settable systems creates no difficulties in defining causal relations, as direct causality is a property solely of the response function on its domain. Moreover, by requiring that settings have more than one admissible value, we ensure that these domains contain at least two

2. We are grateful to two of the referees for emphasizing this.

points, making possible the interventions supporting definitions of effects in settable systems. We will return to this point shortly.

In the PCM, endogenous variables are usually observable, although this is not formally required. Structurally endogenous settable variables X may also be observable or not.

Fortunately, the PCM treats a variable as a background variable or an endogenous variable relative to the analysis. If the effects of a variable are of interest, it can be converted to an endogenous variable in an alternative PCM. Nevertheless, the PCM does not provide guidance on whether to treat a variable as a background variable or an endogenous one. This decision is entirely left to the researcher's discretion. For example, the "disturbances" in the Markovian PCM "represent background variables that the investigator chooses not to include in the analysis" (Pearl, 2000, p. 68), but the PCM does not specify how an investigator chooses to include variables in the analysis. Nor is it clear that background variables are necessary to the analysis in the first place. For example, Dawid (2002, p. 183) states that "when the additional variables are pure mathematical fictions, introduced merely so as to reproduce the desired probabilistic structure of the domain variables, there seems absolutely no good reason to include them in the model."

Settable systems permit but do not require background variables. Further, and of particular significance, in a settable system a governing principle such as optimization provides a formal way to distinguish between fundamental settable variables (exogenous variables) and other settable variables (endogenous variables). In particular, the decision problem determines if a variable is exogenous or endogenous. For instance, in our clamped learning example, the optimal network weights $\tilde{\alpha}^*$ and $\tilde{\beta}^*$ minimize the loss function $L(\alpha, \beta, \gamma)$. On the other hand, although the elements of γ are variables, our learning example does not specify a decision problem that determines how these are generated. This distinction endows the variables $\tilde{\alpha}^*$ and $\tilde{\beta}^*$ with the status of endogenous variables and the elements of γ with the status of structurally exogenous variables.

Thus, carefully and explicitly specifying the decision problems and governing principles in settable systems provides a systematic way to distinguish between exogenous and endogenous variables. This formalizes and extends the distinctions between the PCM endogenous and exogenous variables.

The PCM has been fruitfully applied in the sciences (e.g., Shipley, 2000). Nevertheless, because the PCM is agnostic concerning the status of variables, two researchers may employ two possibly inconsistent PCMs to study the same scientific phenomena. To resolve such inconsistencies, one may use the fact that under suitable assumptions, causal relations imply empirically testable conditional independence relations among system variables (Pearl, 2000; Chalak and White, 2008b). This yields procedures for falsifying causal structures that are inconsistent with data. Such procedures at best identify a class of observationally equivalent causal models, so resolution of inconsistencies by this means is not guaranteed. On the other hand, specifying the decision problems underlying the phenomena of interest may, among other things, offer guidance as to which (if either) model is more suitable to the analysis. The settable systems framework provides the foundation necessary for this in the context of optimally interacting agents under uncertainty. We emphasize that agents and their decision problems may be defined in such a way as to apply even to physical or biological systems not usually thought of in these terms; any system involving optimizing (e.g., least energy, maximum entropy) and/or equilibrium falls into this framework.

3.5 Unclamped Learning and Settable Systems

Now consider how our original unclamped learning example is represented using settable systems. We begin by recognizing that the solution to a given optimization problem need not be unique, but is in general a set. When the solution depends on other variables, the solution is in general a correspondence, not a function (see, e.g., Berge, 1963). Thus, we write the solution to the unclamped learning problem as

$$(\mathbb{A}^*(\gamma), \mathbb{B}^*(\gamma)) := \arg\min_{\alpha,\beta} L(\alpha, \beta, \gamma),$$

where $\mathbb{A}^*(\gamma)$ and $\mathbb{B}^*(\gamma)$ define correspondences.

Due to the linear network architecture, we can explicitly represent $\mathbb{A}^*(\gamma)$ and $\mathbb{B}^*(\gamma)$ as

$$\begin{aligned} \mathbb{A}^*(\gamma) &= \{\alpha : [\mu_{XX} - \mu_X^2](\alpha - \mu_Y) + (\mu_{XY} - \mu_X\mu_Y)\mu_X = 0\}, \\ \mathbb{B}^*(\gamma) &= \{\beta : [\mu_{XX} - \mu_X^2]\beta - (\mu_{XY} - \mu_X\mu_Y) = 0\}. \end{aligned}$$

When $\mu_{XX} - \mu_X^2 > 0$, $\mathbb{A}^*(\gamma)$ and $\mathbb{B}^*(\gamma)$ each have a unique element, namely

$$\begin{aligned} \alpha^* &= \mu_Y - [\mu_{XX} - \mu_X^2]^{-1}(\mu_{XY} - \mu_X\mu_Y)\mu_X, \\ \beta^* &= [\mu_{XX} - \mu_X^2]^{-1}(\mu_{XY} - \mu_X\mu_Y). \end{aligned}$$

When $\mu_{XX} - \mu_X^2 = 0$, we can select a unique value from each of $\mathbb{A}^*(\gamma)$ and $\mathbb{B}^*(\gamma)$. Choosing the simplest representation and the simplest computation of the prediction yields $\alpha^* = \mu_Y$ and $\beta^* = 0$. We thus represent optimal weights using response functions r_1 and r_2 as

$$\begin{aligned} \alpha^* &= r_1(\gamma) := \mu_Y - 1_{\{\mu_{XX} - \mu_X^2 > 0\}}[\mu_{XX} - \mu_X^2]^{-1}(\mu_{XY} - \mu_X\mu_Y)\mu_X, \\ \beta^* &= r_2(\gamma) := 1_{\{\mu_{XX} - \mu_X^2 > 0\}}[\mu_{XX} - \mu_X^2]^{-1}(\mu_{XY} - \mu_X\mu_Y). \end{aligned}$$

These response functions do represent fixed points of the equations in (5). This illustrates the role that fixed points can play in determining the response functions. Observe, however, that we do not require a *unique* fixed point.

Applying the settable system definition of direct causality, we have that a given element of γ, say γ_i, does not directly cause α^* (resp. β^*) if $r_1(\gamma)$ (resp. $r_2(\gamma)$) defines a function constant in γ_i for all admissible values of the other elements of γ. Otherwise, that element is a direct cause of α^* (resp. β^*). Here, each element of γ directly causes both α^* and β^*.

In this example, we have the settable system representation

$$\mathcal{X}_1(0) = r_1(\mathcal{X}_0(1)), \qquad \mathcal{X}_2(0) = r_2(\mathcal{X}_0(1)),$$

where $\mathcal{X}_0(0) := \mathcal{X}_0(1) := \gamma, \mathcal{X}_1(1) := \alpha, \mathcal{X}_2(1) := \beta$ as before, but now $\mathcal{X}_1(0) := \alpha^*$ and $\mathcal{X}_2(0) := \beta^*$.

Finally, we note that the system outputs of the clamped and unclamped systems are *mutually consistent*, in the sense that if we plug the responses of the unclamped system into the response functions $(\tilde{r}_1, \tilde{r}_2)$ of the clamped system as settings, we obtain clamped responses that replicate the responses of the unclamped system. That is, putting $\mathcal{X}_1^c(1) = \mathcal{X}_1^u(0)$ and $\mathcal{X}_2^c(1) = \mathcal{X}_2^u(0)$, where we now employ the superscripts c and u to clearly distinguish clamped and unclamped system settable variables, we have

$$\mathcal{X}_1^u(0) = \tilde{r}_1(\mathcal{X}_2^u(0), \mathcal{X}_0(1)), \qquad \mathcal{X}_2^u(0) = \tilde{r}_2(\mathcal{X}_1^u(0), \mathcal{X}_0(1)),$$

as some simple algebra will verify. This mutual consistency is ensured by the governing principle of optimization.

3.6 Partitioning in Settable Systems

In the PCM, the role of submodels (Pearl, 2000, Def. 7.1.2) is to specify which endogenous variables are subject to manipulation; the do operator specifies which values the manipulated variables take. In settable systems, submodels and the do operator are absent. Nevertheless, settable systems do have an analog of submodels, but instead of specifying which variables are to be manipulated, a settable system specifies which system variables are free to respond to the others. In our learning examples, settable systems specify which variables are unclamped. In our first example, both variables are unclamped. In the second example, the variables are considered one at a time, and each variable is unclamped in turn.

3.6.1 PARTITIONING

A formal mathematical implementation of these specifications is achieved by *partitioning*. Partitioning operates on an index set I whose elements are in one-to-one correspondence to the structurally endogenous (non-fundamental) settable variables. In our learning examples, there are two such variables, so the index set can be chosen to be $I = \{1,2\}$.

Let I be any set with a countable number of elements. A partition Π is a collection of subsets $\Pi_1, \Pi_2, ...$ of I that are mutually exclusive ($\Pi_a \cap \Pi_b = \varnothing, a \neq b$) and exhaustive ($\cup_b \, \Pi_b = I$). Examples are the *elementary partition*, $\Pi^e := \{\Pi^e_1, ..., \Pi^e_n\}$, where $\Pi^e_1 := \{1\}, \Pi^e_2 := \{2\}, ...$, and the *global partition* $\Pi^g := \{\Pi^g_1\}$, where $\Pi^g_1 := I$.

When $I = \{1,2\}$, these are the only two possible partitions: $\Pi^e = \{\Pi^e_1, \Pi^e_2\}$, where $\Pi^e_1 = \{1\}$ and $\Pi^e_2 = \{2\}$; and $\Pi^g = \{\Pi^g_1\}$, where $\Pi^g_1 = \{1,2\}$.

We interpret the partition elements as specifying which of the system variables are jointly free to respond to the remaining variables of the system, according to the governing principle of the system (e.g., optimization). In our machine learning examples with $I = \{1,2\}$, the element $\Pi^e_1 = \{1\}$ of the elementary partition Π^e specifies that variable 1 (i.e., $\tilde{\alpha}^*$) is free to respond to all other variables of the system (i.e., (β, γ)), whereas $\Pi^e_2 = \{2\}$ specifies that variable 2 (i.e., $\tilde{\beta}^*$) is free to respond to all other variables of the system (i.e., (α, γ)). The element $\Pi^g_1 = \{1,2\}$ of the global partition specifies that variables 1 and 2 (i.e., (α^*, β^*)) are jointly free to respond to all other variables of the system (i.e., γ).

In settable systems, response functions are partition specific. With Π^e, we have

$$\tilde{\alpha}^* = \tilde{r}_1(\beta, \gamma), \quad \tilde{\beta}^* = \tilde{r}_2(\alpha, \gamma);$$

with Π^g, we have

$$\alpha^* = r_1(\gamma), \quad \beta^* = r_2(\gamma)$$

for the response functions $(\tilde{r}_1, \tilde{r}_2)$ and (r_1, r_2) defined above. This implies that the settable variables and the resulting causal relations are *partition specific*.

We note that the distinction between the response functions $(\tilde{r}_1, \tilde{r}_2)$ and (r_1, r_2) is not due to additional constraints imposed on the optimization problem per se. Instead, the distinction follows from whether learning occurs with or without clamping and hence on whether or not alpha and beta respond jointly. Thus, different optimization problems yield different corresponding partitions and response functions.

These partitioning concepts and principles extend to systems with any number of structurally endogenous variables. We discuss further examples below.

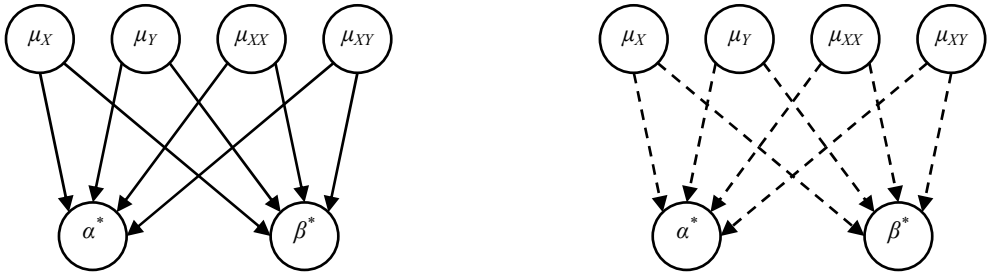


Figure 1: PCM Directed Acyclic Graphs

3.6.2 SETTABLE SYSTEM CAUSAL GRAPHS

Given the applicability of the PCM to the unclamped learning example, this system has an associated PCM directed acyclic graph (DAG). The particular representation of this graph depends on whether or not the background variables are observable or not. Figure 1(a) depicts the case of observable γ and Figure 1(b) that of unobservable γ. The solid arrows in Figure 1(a) indicate that the background variables are observable, whereas the dashed arrows in Figure 1(b) indicate that the background variables are not observable.

In interpreting these graphs, note that the arrows, whether solid or dashed, represent the functional relationships present. They do not, however, represent causal relations, as in the PCM these are defined to hold only between endogenous variables, and no arrows link the endogenous variables here. Pearl (2000) often uses the term "influence" to refer to situations involving functional dependence, but not causal dependence. In this sense, the arrows in these DAGs represent "influences."

In contrast, due to the lack of a fixed point, the PCM does not apply to the learning with clamping example. Necessarily, the PCM cannot supply a causal graph.

In settable systems, partitions play a key role in constructing causal graphs that represent direct causality relations. To see how, consider our clamped learning example. Here, μ_Y (i.e., $\mathcal{X}_{0,2}(1)$) does not directly cause $\tilde{\beta}^*$ ($\mathcal{X}_2(0)$), whereas $\mu_X, \mu_{XX}, \mu_{XY}$, and α ($\mathcal{X}_{0,1}(1), \mathcal{X}_{0,3}(1), \mathcal{X}_{0,4}(1)$, and $\mathcal{X}_1(1)$) are direct causes of $\tilde{\beta}^*$ ($\mathcal{X}_2(0)$). We can succinctly and unambiguously state these causal relations in terms of settable variables by saying that $\mathcal{X}_{0,2}$ does not directly cause $\mathcal{X}_2$, whereas $\mathcal{X}_{0,1}, \mathcal{X}_{0,3}, \mathcal{X}_{0,4}$, and $\mathcal{X}_1$ are direct causes of $\mathcal{X}_2$.

For each block Π_b of a partition $\Pi = \{\Pi_b\}$, we construct a settable system causal graph by letting nodes correspond to settable variables. If one settable variable directly causes another, we draw an arrow from the node representing the cause to that representing the responding settable variable. Note that in contrast to the DAGs for the PCM, we represent all direct causal links as solid arrows, letting dashed nodes represent unobservable settable variables. The motivation for this is that unobservability is a property of the settable variable (the node), not the links between nodes.

Figures 2(a) and 2(b) depict the causal graphs for our clamped learning example. There are two causal graphs, as the clamped learning example expresses the elementary partition $\Pi^e = \{\{1\}, \{2\}\}$. For purposes of illustration, we depict the case in which γ is unobserved.

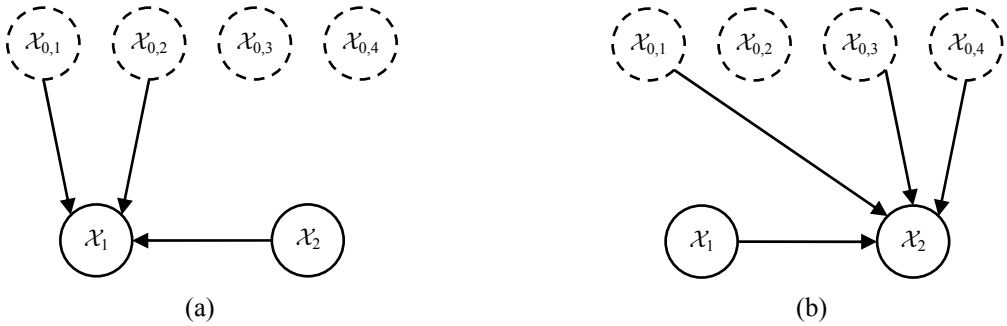


Figure 2: Block-specific Settable System Causal Graphs for the Elementary Partition

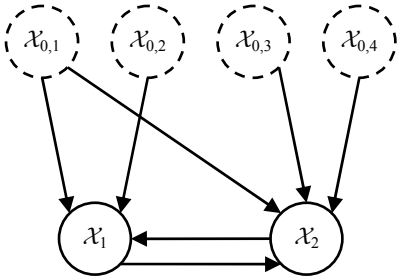


Figure 3: Settable System Superimposed Causal Graph for the Elementary Partition

For convenience, we may superimpose settable system causal graphs. Superimposing Figures 2(a) and 2(b) gives Figure 3. This is a cyclic graph. Nevertheless, this cyclicality does not represent true simultaneity; it is instead an artifact of the superimposition.

The settable system causal graph for the global partition $\Pi^g = \{\{1,2\}\}$ representing unclamped learning is depicted in Figure 4. Observe that this reproduces the connectivity of Figure 1. Note that in Figure 4, the nodes represent settable variables and the arrows represent direct causes. In Figure 1, the nodes represent background or endogenous variables and the arrows represent non-causal "influences."

We emphasize that the causal graphs associated with settable systems are not necessary to the analysis. Rather, they are sometimes helpful in succinctly representing and studying causal relations.

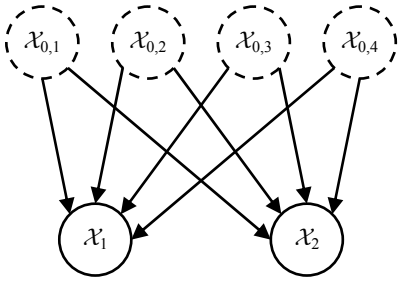


Figure 4: Settable System Causal Graph for the Global Partition

3.7 Further Examples Motivating Settable System Features

We now introduce two further features of settable systems, countable dimension and attributes, using examples involving machine learning algorithms and networks with hidden units. This provides further interesting contrasts between settable systems and the PCM.

3.7.1 A MACHINE LEARNING ALGORITHM AND COUNTABLE DIMENSIONALITY

So far, we have restricted attention to the optimal network weights for linear least-squares machine learning. Now consider the machine learning algorithm itself. For this, let

$$\hat{\mu}_{x,0} = \hat{\mu}_{y,0} = \hat{\mu}_{xx,0} = \hat{\mu}_{xy,0} = \hat{\alpha}_0 = \hat{\beta}_0 = 0,$$

and perform the recursion

$$\begin{aligned}
\hat{\mu}_{x,n} &= \hat{\mu}_{x,n-1} + n^{-1}(x_n - \hat{\mu}_{x,n-1}) \\
\hat{\mu}_{y,n} &= \hat{\mu}_{y,n-1} + n^{-1}(y_n - \hat{\mu}_{y,n-1}) \\
\hat{\mu}_{xx,n} &= \hat{\mu}_{xx,n-1} + n^{-1}(x_n^2 - \hat{\mu}_{xx,n-1}) \\
\hat{\mu}_{xy,n} &= \hat{\mu}_{xy,n-1} + n^{-1}(x_n y_n - \hat{\mu}_{xy,n-1}) \\
\hat{\beta}_n &= 1_{\{\hat{\mu}_{xx,n} - \hat{\mu}_{x,n}^2 > 0\}}[\hat{\mu}_{xx,n} - \hat{\mu}_{x,n}^2]^{-1}(\hat{\mu}_{xy,n} - \hat{\mu}_{x,n}\hat{\mu}_{y,n}) \\
\hat{\alpha}_n &= \hat{\mu}_{y,n} - \hat{\beta}_n \hat{\mu}_{x,n}, \quad n = 1, 2, ...
\end{aligned} \tag{9}$$

Variables determined outside the system are the observed data sequences $x := (x_1, x_2, ...)$ and $y := (y_1, y_2, ...)$. Variables determined within the system are $\hat{\mu}_x := (\hat{\mu}_{x,0}, \hat{\mu}_{x,1}, ...), \hat{\mu}_y := (\hat{\mu}_{y,0}, \hat{\mu}_{y,1}, ...), \hat{\mu}_{xx} := (\hat{\mu}_{xx,0}, \hat{\mu}_{xx,1}, ...), \hat{\mu}_{xy} := (\hat{\mu}_{xy,0}, \hat{\mu}_{xy,1}, ...), \hat{\alpha} = (\hat{\alpha}_0, \hat{\alpha}_1, ...)$, and $\hat{\beta} := (\hat{\beta}_0, \hat{\beta}_1, ...)$. Under mild conditions, $\hat{\alpha}_n$ converges to α^* and $\hat{\beta}_n$ converges to β^*.

We now ask whether this system falls into the PCM. The answer is no, because the PCM requires the dimensions of the background and endogenous variables to be finite. Here these dimensions are countably infinite. The PCM does not apply. (As a referee notes, however, a countably infinite version of the PCM has recently been discussed by Eichler and Didelez, 2007).

In contrast, settable systems encompass this learning system by permitting the settable variables to be of countably infinite dimension. The definitions of direct causality and the notion of partitioning operate identically in either the finite or the countably infinite case. Settable systems generally accommodate any recursive learning algorithm involving data sequences of arbitrary length.

3.7.2 LEARNING WITH A HIDDEN UNIT NETWORK AND ATTRIBUTES

To motivate the next feature of settable systems, we return to considering the effect on an optimal network weight of interventions to distributional parameters, γ, and another network weight. Now, however, we modify the prediction function to be that defined by

$$f(X; \alpha, \beta) = \alpha\, \phi(\beta X).$$

This is a single hidden layer feedforward network with a single hidden unit having the activation function ϕ. For concreteness, let ϕ be the standard normal density. This activation function often appears in radial basis function networks. For clarity, we consider only a single input X, a single

input-to-hidden weight β, and a single hidden-to-output weight α. This elementary structure suffices to make our key points and keeps our notation simple.

Now the expected squared prediction error is

$$L(\alpha,\beta,\gamma;\phi) := E_\gamma([Y-\alpha\,\phi(\beta X)]^2).$$

Here, γ reverts to representing the general parameter indexing F_γ. The choice $\gamma := (\mu_X,\mu_Y,\mu_{XX},\mu_{XY})$ considered above is no longer appropriate, due to the nonlinearity in network output induced by ϕ. Further, note the presence of the hidden unit activation function ϕ in the argument list of L. We make this explicit, as ϕ certainly helps determine prediction performance, and it has a key role to play in our subsequent discussion.

Now consider the clamped optimization problems corresponding to the elementary partition Π^e. This yields solutions

$$\begin{aligned}\widetilde{\mathbb{A}}^*(\beta,\gamma;\phi) &:= \arg\min_{\alpha\in\mathbb{A}} L(\alpha,\beta,\gamma;\phi),\\ \widetilde{\mathbb{B}}^*(\alpha,\gamma;\phi) &:= \arg\min_{\beta\in\mathbb{B}} L(\alpha,\beta,\gamma;\phi).\end{aligned}$$

We ensure the existence of compact-valued correspondences $\widetilde{\mathbb{A}}^*(\beta,\gamma;\phi)$ and $\widetilde{\mathbb{B}}^*(\alpha,\gamma;\phi)$ by (among other things) taking $\mathbb{A}$ and $\mathbb{B}$ to be compact subsets of $\mathbb{R}$. Elements $\tilde\alpha^*$ of $\widetilde{\mathbb{A}}^*(\beta,\gamma;\phi)$ and $\tilde\beta^*$ of $\widetilde{\mathbb{B}}^*(\alpha,\gamma;\phi)$ satisfy the necessary first order conditions

$$\begin{aligned}E_\gamma([\phi(\beta X)]^2)\tilde\alpha^* - E_\gamma(\phi(\beta X)Y) &= 0,\\ E_\gamma(D\phi(\tilde\beta^* X)Y) - \alpha E_\gamma[D\phi(\tilde\beta^* X)\phi(\tilde\beta^* X)] &= 0,\end{aligned}$$

where $D\phi$ denotes the first derivative of the ϕ function. We caution that although these relations necessarily hold for elements of $\widetilde{\mathbb{A}}^*(\beta,\gamma;\phi)$ and $\widetilde{\mathbb{B}}^*(\alpha,\gamma;\phi)$, not all (α,β) values jointly satisfying these implicit equations are members of $\widetilde{\mathbb{A}}^*(\beta,\gamma;\phi)$ and $\widetilde{\mathbb{B}}^*(\alpha,\gamma;\phi)$. Some solutions to these equations may be local minima, inflection points, or (local) maxima.

The PCM does not apply here, due (among other things) to the absence of a unique fixed point. Nevertheless, settable systems do apply, using a principled selection of elements from $\widetilde{\mathbb{A}}^*(\beta,\gamma;\phi)$ and $\widetilde{\mathbb{B}}^*(\alpha,\gamma;\phi)$, respectively. We write these selections

$$\tilde\alpha^* = \tilde r_1(\beta,\gamma;\phi), \quad \tilde\beta^* = \tilde r_2(\alpha,\gamma;\phi).$$

The feature distinguishing this example from our earlier examples is the appearance in the response functions of the hidden unit activation function ϕ. The key feature of ϕ is that it takes one and only one value: ϕ is the standard normal density. It is therefore not a variable. Consequently, it cannot be a setting, and it is distinct from any of the other objects we have previously examined. We define an *attribute* to be any object specified a priori that helps determine responses but is not variable. We associate attributes with the system units. Any attribute of the system itself is a *system attribute*; we formally associate system attributes to each system unit. Here, ϕ is a system attribute. Because a unit's associated attributes are constant, they are not subject to counterfactual variation. Nevertheless, attributes may differ across units.

One generally useful attribute is the attribute of *identity*. This is a label assigned to each unit of a given system that can take only the assigned value, and whose value is shared by no other unit

of the system. The identity attribute is required by settable systems, as the identity labels are those explicitly used in the partitioning operation. The identity attribute is also a feature of the PCM, as background and endogenous variables are distinct types of objects, and elements of each distinct type have identifying subscripts.

When attributes beyond identity are present, they need not be distinct across units. For example, the quantity n appearing in several of the response functions in the learning algorithm of Equation (9) is an attribute shared by those units.

We emphasize that attributes are relative to the particular structural system, not somehow absolute. Some objects may be taken as attributes solely for convenience. For example, one might consider several different possible activation functions and attempt to learn the best one for a given problem. In such systems, the hidden unit activation is no longer an attribute but is an endogenous variable. In other cases, it may be more convenient to treat the activation function as hard-wired, in which case the activation function is an attribute. Indeed, any hard-wired aspect of the system is properly an attribute. Convenience may even dictate treating as attributes objects that are in principle variable, but whose degree of variation is small relative to that of other system variables of interest.

Other system aspects are more inherently attributes. Because of their fundamental role and their invariance, such attributes are easily taken for granted and thus overlooked. Our least-squares learning example is a case in point. Specifically, the loss function itself is properly viewed as an attribute.

A useful way to appreciate this is to consider the loss functions

$$L_p(\alpha,\beta,\gamma) := \int |y - f(x;\alpha,\beta)|^p dF_\gamma(x,y), \quad p > 0.$$

In our examples so far, we always take $p = 2$, so $L = L_2$. Different choices are possible, yielding different loss functions. A leading example is the choice $p = 1$. Whereas $p = 2$ yields forecasts that approximate the conditional mean of Y given X, $p = 1$ yields forecasts that approximate the conditional median of Y given X.

Because p is a constant specified a priori and because p helps determine the optimal responses, p is an attribute. When the forecaster's goal is explicitly to provide a forecast based on the conditional mean, it makes little sense to consider values of p other than 2, because no other value of p is guaranteed to generally deliver an approximation to the conditional expectation. Put somewhat differently, it may not make much sense to attempt to endogenize p and choose an "optimal" value of p from some set of admissible values because the result of choosing different values for p is to modify the very goal of the learning exercise. Nor can one escape from attributes by endogenizing p; as long as there is some optimality criterion at work, this criterion is properly an attribute of the system.

Another important example of inherent attributes is provided by the sets $\mathbb{S}_i$ that specify the admissible values taken by the settings $\mathcal{X}_i(1)$ and responses $\mathcal{X}_i(0)$. These are properly specified a priori; they take one and only one value for each unit i; and they play a fundamental role in determining system responses.

Because attributes in settable systems are fixed a priori for a given unit, they take values in a (non-empty) degenerate set. Accordingly, attributes cannot be settings, and thus can never be potential causes, much less direct causes. This formal distinction between attributes and potential causes is unambiguous in settable systems.

3.7.3 ATTRIBUTES IN THE PCM

In contrast, a somewhat ambiguous situation prevails in the PCM. Viewing attributes as a subset of those objects having no causal status, Pearl (2000, p. 98) states that attributes can be treated as elements of u, the background variables.[3] This overlooks the key property we wish to assign to attributes: for a given unit, they are fixed, not variable. Such objects thus cannot belong to u if one takes the word "variable" at face value. In our view, assigning attributes to u misses the opportunity to make an important distinction between invariant aspects of the system units on the one hand and counterfactual variation admissible for the system unit values on the other. Among other things, assigning attributes to u interferes with assigning natural causal roles to structurally exogenous variables.

Further, just as for endogenous and exogenous variables, the PCM does not provide guidance about how to select attributes. In contrast, settable systems clearly identify attributes as invariant features of the system units that embody fundamental aspects of the decision problem represented by the settable system.

Below, we will further distinguish attributes from variables when we discuss stochastic settable systems.

3.8 A Comparative Review of PCM and Settable System Features

At this point, it is helpful to take stock of the features of settable systems that we have so far introduced and contrast these with corresponding features of the PCM.

(1) Settable systems explicitly represent the dual roles of the variables of structural systems using settable variables. These dual roles are present but implicit in the PCM. Settable variables can be responses, or they can be set to given values (settings). The explicit representation of these dual roles in settable systems makes possible implicitly defined structural relations that may not be representable in the PCM. Further, these implicit structural relations may involve correspondences rather than functions. Principled selections from these correspondences yield unique response functions in settable systems.

(2) In settable systems, all variables of the system, structurally exogenous or endogenous, have causal status, in that they can be potential causes or direct causes. Further, no assumptions are made as to the observability of system variables: structurally exogenous variables may be either observable or unobservable; the same is true for structurally endogenous variables. In particular, this permits settable systems to admit unobserved causes and results in causal relations that are not relative to an observer. In contrast, the PCM admits causal status only for endogenous variables. For the PCM, structurally exogenous unobserved causes are ruled out. Although the PCM does permit treating background variables as endogenous variables in alternative systems, it is silent as to how to distinguish between exogenous and endogenous variables. On the other hand, the governing principles in settable systems provide a formal and explicit means for distinguishing between endogenous and exogenous variables.

(3) Settable systems admit straightforward definitions of interventions and direct effects. These notions, while present, are less direct in the formal PCM.

(4) In settable systems, partitioning permits specification of different mutually consistent versions of a given structural system in which different groups of variables are jointly free to respond to the other variables of the system. In particular, system variables can respond either singly or jointly

3. This possibility is also suggested by two referees.

to the other variables of the system, as illustrated by our examples of learning with or without clamping. Similar exercises are possible in the PCM using submodels and the do operator, but the PCM requirement of a unique fixed point limits its applicability. Specifically, we saw that learning with clamping falls outside the PCM. Halpern's (2000) GPCM does apply to such systems, but causal discourse is problematic, due to the absence of the potential response function. In settable systems, fixed points are not required, and causal notions obtain without requiring the potential response function. This permits settable systems to provide causal meaning in our examples of learning with or without clamping.

(5) Settable systems can have a countable infinity of units, whereas the PCM requires a finite number of units.

(6) In settable systems, attributes are a priori constants associated with the units that help determine responses. In the PCM, attributes are not necessarily linked to the system units. Further, they are treated not as constants, but as background variables, resulting in potential ambiguity. The PCM is silent as to how to distinguish between attributes and variables.

Some features of settable systems, such as relaxing the assumption of unique fixed points (point 4) and accommodating an infinity of agents (point 5), are entirely unavailable in the PCM. The remaining settable systems features above rigorously formalize and extend or refine related PCM features and thus permit more explicit causal inference.

4. Stochastic Settable Systems: Heuristics

In the PCM, randomness does not formally appear until definition 7.1.6 (*probabilistic causal model*). Nevertheless, Pearl's (2000) definitions 7.1.1 through 7.1.5 (*causal model*, *submodel*, *effect of action*, *potential response,* and *counterfactual*) explicitly refer to "realizations" *pa* or *x* of endogenous variables *PA* or X. These references make sense only if *PA* and X are interpreted as random vectors. Although *u* is not explicitly called a realization, the language of definition 7.1.1 further suggests that *u* is viewed as a realization of random background variables, U. This becomes explicit in definition 7.1.6, where PCM background variables U become governed by a probability measure *P*. Randomness of endogenous variables is then induced by their dependence on U. In this sense, definitions 7.1.1 through 7.1.5 do not have fully defined content until definition 7.1.6 resolves the meaning of U, V, PA, and X. Nevertheless, definitions 7.1.1 through 7.1.5 are perfectly meaningful, simply viewing the referenced variables as real numbers.

The settable systems discussed so far are entirely non-stochastic: the settings and responses defined in Section 3 are real numbers, not random variables. Nevertheless, we can connect causal and stochastic structures in settable systems by viewing settings and responses as realizations of random variables, in much the same spirit as the PCM. In this section we discuss some specifics of this connection.

4.1 Introducing Randomness into Settable Systems

First, instead of only the background variables *u* representing realizations of random variables, in settable systems *all* settings represent realizations of random variables governed by an underlying probability measure. For example, in our hidden unit clamped learning example, (α, β, γ) are realizations of random variables (A, B, C) governed by a probability measure P on an underlying measurable space $(\Omega, \mathcal{F})$. The randomness of the responses is induced by their dependence on the

settings. Thus, in the hidden unit clamped learning example, we have random responses

$$\tilde{A}^* = \tilde{r}_1(B,C;\phi), \qquad \tilde{B}^* = \tilde{r}_2(A,C;\phi).$$

Second, the underlying probability measure for settable systems can depend on the attribute vector, call it $\mathbf{a}$, of the system. Whereas in the PCM attributes are "lumped together" with other background variables, and may therefore be random, this is not permitted in settable systems. In settable systems, attributes are specified a priori and take one and only one value, $\mathbf{a}$. Because of its a priori status, this value is non-random.

It follows that the probability measure governing the settable system can be indexed by $\mathbf{a}$. This is not an empty possibility; it has clear practical value. One context in which this practical value arises is when attention focuses only on the units of some subsystem of a larger system. For example, consider the least squares machine learning algorithm of the equations in (9), and focus attention on the subsystem

$$\begin{aligned} \hat{B} &= 1_{\{\hat{M}_{xx}-\hat{M}_x^2>0\}}[\hat{M}_{xx}-\hat{M}_x^2]^{-1}(\hat{M}_{xy}-\hat{M}_x\hat{M}_y), \\ \hat{A} &= \hat{M}_y - \hat{B}\hat{M}_x. \end{aligned}$$

Note that we have modified the notation to reflect the fact that the settings $\hat{M}_x, \hat{M}_y, \hat{M}_{xx}$, and $\hat{M}_{xy}$ are now random variables. These generate realizations $\hat{\mu}_{x,n}, \hat{\mu}_{y,n}, \hat{\mu}_{xx,n}$, and $\hat{\mu}_{xy,n}$ under a probability measure P_n, which is that induced by the probability measure governing the random fundamental settings $\{(X_1,Y_1),...,(X_n,Y_n)\}$. Note the explicit dependence of the probability measure P_n on the attribute n. The fact that this probability measure can depend on attributes underscores their nature as a priori constants in settable systems.

4.2 Some Formal Properties of Stochastic Settable Systems

Given attributes $\mathbf{a}$, we let $(\Omega, \mathcal{F}, P_\mathbf{a})$ denote the complete probability space on which the settings and responses are defined. Here, Ω is a set (the "universe") whose elements ω index possible outcomes ("possibilities"); $\mathcal{F}$ is a $\sigma-$field of measurable subsets of Ω whose elements represent events; and $P_\mathbf{a}$ is a probability measure (indexed by $\mathbf{a}$) on the measurable space $(\Omega, \mathcal{F})$ that assigns a number $P_\mathbf{a}(F)$ to each event $F \in \mathcal{F}$. See, for example, White (2001, Chap. 3) for an introductory discussion of measurable spaces and probability spaces.

We decompose ω as $\omega := (\omega_r, \omega_s)$, with $\omega_r \in \Omega_r, \omega_s \in \Omega_s$, so that $\Omega = \Omega_r \times \Omega_s$. As we discuss next, this enables distinct components of ω to underlie responses (ω_r) and settings (ω_s). This facilitates straightforward and rigorous definitions of counterfactuals and interventions. These notions in turn support a definition of direct effect.

To motivate the foundations for defining counterfactuals, again consider the hidden unit clamped learning example. Formally, the random settings (A,B,C) are measurable functions $A : \Omega_s \to \mathbb{R}$, $B : \Omega_s \to \mathbb{R}$, and $C : \Omega_s \to \mathbb{R}^m$, $m \in \mathbb{N}$. Letting ω_s belong to Ω_s, we take the setting values to be the realizations

$$\begin{aligned} \alpha &= A(\omega_s) =: \mathcal{X}_1(\omega,1), \\ \beta &= B(\omega_s) =: \mathcal{X}_2(\omega,1), \\ \gamma &= C(\omega_s) =: \mathcal{X}_0(\omega,1). \end{aligned}$$

Observe that the settings depend only on the ω_s component of ω. We make this explicit in $A(\omega_s)$, $B(\omega_s)$, and $C(\omega_s)$, but leave this implicit in writing $\mathcal{X}_0(\omega,1)$, $\mathcal{X}_1(\omega,1)$, and $\mathcal{X}_2(\omega,1)$ for notational convenience.

The responses are determined similarly:

$$\begin{aligned}\tilde{A}^*(\omega) &= \tilde{r}_1(B(\omega_s),C(\omega_s),\omega_r;\phi) = \tilde{r}_1(\mathcal{X}_2(\omega,1),\mathcal{X}_0(\omega,1),\omega_r;\phi) =: \mathcal{X}_1(\omega,0),\\ \tilde{B}^*(\omega) &= \tilde{r}_2(A(\omega_s),C(\omega_s),\omega_r;\phi) = \tilde{r}_2(\mathcal{X}_1(\omega,1),\mathcal{X}_0(\omega,1),\omega_r;\phi) =: \mathcal{X}_2(\omega,0).\end{aligned}$$

Note that we now make explicit the possibility that the response functions may depend directly on ω_r. This dependence was absent in all our previous examples but is often useful in applications, as this dependence permits responses to embody an aspect of "pure" randomness. From now on, we will include ω_r as an explicit argument of the response functions.

In the deterministic systems previously considered, we viewed the fundamental setting $\mathcal{X}_0(1)$ as a primitive object and adopted the convention that $\mathcal{X}_0(0) := \mathcal{X}_0(1)$. Once settings and responses depend on ω, it becomes necessary to modify our conventions regarding the fundamental settable variables $\mathcal{X}_0$, as $\mathcal{X}_0$ is no longer determined outside the system. The role of the system primitive is now played by ω, the *primary setting*. We represent this as the settable variable defined by $\mathcal{X}_*(\omega,0) := \mathcal{X}_*(\omega,1) := \omega$. We now view $\mathcal{X}_0(\omega,0)$ as a response to ω_s and we take $\mathcal{X}_0(\cdot,1) := \mathcal{X}_0(\cdot,0)$.

In the current stochastic framework, the feature that distinguishes $\mathcal{X}_0$ from other settable variables is that the response $\mathcal{X}_0(\omega,0)$ depends only on ω_s, whereas responses of other settable variables can depend directly on other settings and on ω_r. Given the availability of $\mathcal{X}_*$, there is no guarantee or requirement that such a settable variable $\mathcal{X}_0$ exists. Nevertheless, such *fundamental stochastic settable variables* $\mathcal{X}_0$ are often an important and useful feature in applications, as our machine learning examples demonstrate.

The definition of direct causality in stochastic settable systems is closely parallel to that in the non-stochastic case. Specifically, consider the partition $\Pi = \{\Pi_b\}$, and suppose i belongs to the partition element Π_b. Let $\mathcal{X}_{(b)}(\omega,1)$ denote setting values for the settable variables whose indexes do not belong to Π_b, together with the settings $\mathcal{X}_0(\omega,1)$. Then the response $\mathcal{X}_i(\omega,0)$ is given by

$$\mathcal{X}_i(\omega,0) := r_i(\mathcal{X}_{(b)}(\omega,1),\omega_r;\mathbf{a}) = r_i(z_{(b)},\omega_r;\mathbf{a}),$$

where r_i is the associated response function, $\mathbf{a}$ is the attribute vector, and for convenience we write $z_{(b)} := \mathcal{X}_{(b)}(\omega_s,1)$. Then we say that $\mathcal{X}_j$ *does not directly cause* $\mathcal{X}_i$ if $r_i(z_{(b)},\omega_r;\mathbf{a})$ defines a function constant in the element z_j of $(z_{(b)},\omega_r)$ for all values of the other elements of $(z_{(b)},\omega_r)$. Otherwise, we say that $\mathcal{X}_j$ *directly causes* $\mathcal{X}_i$. Thus, $\mathcal{X}_*$ can directly cause $\mathcal{X}_i$; for this, take $z_j = \omega_r$. If $\mathcal{X}_0(\omega,0)$ does not define a constant function of ω_s, we also say that $\mathcal{X}_*$ directly causes $\mathcal{X}_0$. As always, direct causality is relative to the specified partition.

4.3 Counterfactuals, Interventions, and Direct Effects

We now have the foundation necessary to specify "counterfactuals." We begin by defining what is meant by "factual." Suppose for now that all setting and response values apart from ω_r are observable. Specifically, suppose we have realizations of setting values (β,γ) and response value $\tilde{\alpha}^* = \tilde{r}_1(\beta,\gamma,\omega_r;\phi)$, and that ω is such that $\beta = B(\omega_s)$, $\gamma = C(\omega_s)$, and $\tilde{\alpha}^* = \tilde{A}^*(\omega)$, where

$$\tilde{A}^*(\omega) = \tilde{r}_1(B(\omega_s),C(\omega_s),\omega_r;\phi).$$

Then we say that $(\tilde{\alpha}^*, \beta, \gamma)$ are *factual* and that $\omega = (\omega_r, \omega_s)$ is *factual*. Otherwise, we say that $(\tilde{\alpha}^*, \beta, \gamma)$ and ω are *counterfactual*. Specifically, if the realization $(\tilde{\alpha}^*, \beta, \gamma)$ does not obtain, then we say that $(\tilde{\alpha}^*, \beta, \gamma)$ is counterfactual, whereas if we have the realizations $(\tilde{\alpha}^*, \beta, \gamma)$, but ω is such that $\beta \neq B(\omega_s)$, $\gamma \neq C(\omega_s)$, or $\tilde{\alpha}^* \neq \tilde{A}^*(\omega)$ then we say that ω is counterfactual.

There need not be a unique factual ω since it is possible that multiple ω's yield the same realizations of random variables; this creates no logical or conceptual difficulties. Also, we need not observe all settings and responses; an observable subset of these may be factual or counterfactual. To the extent that a given ω generates realizations compatible with factual observables, it may also be viewed as factual to that degree. An ω generating realizations incompatible with factual observables is necessarily counterfactual.

In non-stochastic settable systems, we defined an intervention to a settable variable as a pair of distinct admissible setting values for that settable variable. For example, $\alpha_1 \to \alpha_2 := (\alpha_1, \alpha_2)$. In stochastic settable systems, we express interventions similarly. Specifically, again consider the partition $\Pi = \{\Pi_b\}$, and suppose i belongs to the partition element Π_b, so that

$$\mathcal{X}_i(\omega, 0) := r_i(\mathcal{X}_{(b)}(\omega, 1), \omega_r; \mathbf{a}) = r_i(z_{(b)}, \omega_r; \mathbf{a}).$$

Then an *intervention* $(z_{(b),1}, \omega_{r,1}) \to (z_{(b),2}, \omega_{r,2})$ is a pair $((z_{(b),1}, \omega_{r,1}), (z_{(b),2}, \omega_{r,2}))$ whose elements are admissible and distinct.

Interventions necessarily involve counterfactuals: at most, only one setting can be factual; and for an intervention to be well defined, the other setting value must be distinct. We note that the notion of counterfactuals is helpful mainly for describing interventions. Although our definitions of causality, interventions, or, as we see next, direct effects implicitly involve counterfactuals, they do not formally require this notion.

The *direct effect on $\mathcal{X}_i$ of the intervention* $(z_{(b),1}, \omega_{r,1}) \to (z_{(b),2}, \omega_{r,2})$ is the associated response difference

$$r_i(z_{(b),2}, \omega_{r,2}; \mathbf{a}) - r_i(z_{(b),1}, \omega_{r,1}; \mathbf{a}).$$

Our definitions of interventions and direct effects permit *ceteris paribus* interventions and direct effects. For these, only some finite number (e.g., one) of the settings differs between $(z_{(b),1}, \omega_{r,1})$ and $(z_{(b),2}, \omega_{r,2})$; the other elements are "held constant."

Under suitable conditions (specifically, that the settings $\mathcal{X}_{(b)}(\cdot, 1)$ are an "onto" function), the interventions $(z_{(b),1}, \omega_{r,1}) \to (z_{(b),2}, \omega_{r,2})$ can be equivalently represented as a *primary intervention*

$$\omega_1 \to \omega_2 := (\omega_1, \omega_2) = ((\omega_{r,1}, \omega_{s,1}), (\omega_{r,2}, \omega_{s,2})).$$

That is, primary interventions are pairs (ω_1, ω_2) of elements of Ω. This representation is ensured by specifying that $\omega = (\omega_r, \omega_s)$, permitting ω_r and ω_s to be variation free (i.e., ω_r can vary without inducing any necessary variation in ω_s, and vice versa).

Primary interventions yield a definition of *total effect* as a response difference. In our hidden unit clamped learning example, the total effect on $\mathcal{X}_1$ of $\omega_1 \to \omega_2$ is

$$\begin{aligned} \Delta\mathcal{X}_1(\omega_1, \omega_2, 0) \ &: = \mathcal{X}_1(\omega_2, 0) - \mathcal{X}_1(\omega_1, 0) \\ &= \ \tilde{r}_1(B(\omega_{s,2}), C(\omega_{s,2}), \omega_{r,2}; \phi) - \tilde{r}_1(B(\omega_{s,1}), C(\omega_{s,1}), \omega_{r,1}; \phi). \end{aligned}$$

This is also the direct effect on $\mathcal{X}_1$ of $(Z_{(1)}(\omega_{s,1}), \omega_{r,1}) \to (Z_{(1)}(\omega_{s,2}), \omega_{r,2})$. We emphasize that these effects are, as always, relative to the governing partition. The total effect above is relative to the elementary partition, corresponding to clamped learning.

4.4 Review of Stochastic Settable System Features

In stochastic settable systems, all settings are governed by the underlying probability measure, whereas in the PCM, only the background variables are subject to random variation. Because of their status as a priori constants, attributes can index the settable system probability measure. Stochastic settable systems distinguish between primary settings and, when they exist, fundamental settable variables. Responses may contain an element of pure randomness. The structure of stochastic settable systems also supports straightforward rigorous definitions of direct causes, counterfactuals, interventions, direct effects, and total effects.

5. Stochastic Settable Systems: A Formal Definition

In Sections 3 and 4, we motivated the features of settable systems using a series of closely related machine learning examples. Here we integrate these features to provide a rigorous formal definition of a stochastic settable system $\mathcal{S}^{\Pi} := \{(\mathbf{A},\mathbf{a}),(\Omega,\mathcal{F},P_{\mathbf{a}}),(\Pi,\mathcal{X}^{\Pi})\}$.

To give a concise definition, we first introduce some convenient notation. We write the positive integers as $\mathbb{N}^+$; we also write $\bar{\mathbb{N}}^+ = \mathbb{N}^+ \cup \{\infty\}$. When $n = \infty$, we interpret $i = 1,...,n$ as $i = 1,2,....$ We also write $\mathbb{N} := \{0\} \cup \mathbb{N}^+$, and $\bar{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$. When $m = 0$, we interpret $k = 1,...,m$ as being omitted; thus, when $m = 0$, terms like $\times_{k=1}^{m} A$ or $\times_{k=1}^{m} \mathbb{S}_{0,k}$ are ignored. The notation $\#\Pi$ denotes the number of elements (the cardinality) of the set Π.

Definition 1 (Stochastic Settable System) *Let $n \in \bar{\mathbb{N}}^+$, and let the* unit attribute space *A be a non-empty set. For each* unit *$i = 1,...,n$, let a* unit attribute *a_i be a fixed element of A, such that a_i includes a component of* admissible settings *$\mathbb{S}_i$, a multi-element Borel-measurable subset of $\mathbb{R}$.*

Let $m \in \bar{\mathbb{N}}$. For each $k = 1,...,m$, let a fundamental unit attribute *$a_{0,k}$ be a fixed element of A, such that $a_{0,k}$ includes a component of* admissible fundamental settings *$\mathbb{S}_{0,k}$, a multi-element Borel-measurable subset of $\mathbb{R}$. Write $a_0 := (a_{0,1},...,a_{0,m})$ and $\mathbf{a} := (a_0,a_1,...,a_n) \in \mathbf{A} := (\times_{k=1}^{m} A) \times (\times_{i=1}^{n} A)$, the* joint attribute space.

Let $(\Omega_r,\mathcal{F}_r)$ and $(\Omega_s,\mathcal{F}_s)$ be measurable spaces such that Ω_r and Ω_s each contain at least two elements, and let $(\Omega,\mathcal{F},P_{\mathbf{a}})$ be a complete probability space, where $\Omega := \Omega_r \times \Omega_s$, $\mathcal{F} := \mathcal{F}_r \otimes \mathcal{F}_s$, and $P_{\mathbf{a}}$ is a probability measure indexed by $\mathbf{a} \in \mathbf{A}$.

For each $k = 1,...,m$, let a fundamental response *$Y_{0,k} : \Omega_s \to \mathbb{S}_{0,k}$ be a measurable function and let the corresponding* fundamental setting *be $Z_{0,k} := Y_{0,k}$. Write fundamental settings and responses as $Z_0 := (Z_{0,1},...,Z_{0,m})$ and $Y_0 = Z_0$.*

Let $\Pi = \{\Pi_b\}$ be a partition of $\{1,...,n\}$, with $B := \#\Pi \in \bar{\mathbb{N}}^+$, let $\ell_b := \#\Pi_b$, and let $\mathbf{a}$ determine the multi-element Borel measurable set $\mathbb{S}^{\Pi}_{(b)}(\mathbf{a}) \subset \times_{j \notin \Pi_b} \mathbb{S}_j \times_{k=1}^{m} \mathbb{S}_{0,k}$, $b = 1,...,B$. Suppose there exist measurable functions called settings, *$Z_i^{\Pi} : \Omega_s \to \mathbb{S}_i, i = 1,\ldots,n$, measurable functions called* responses, *$Y_i^{\Pi} : \Omega \to \mathbb{S}_i, i = 1,\ldots,n$, and measurable functions called* joint response functions,

$$r^{\Pi}_{[b]}(\,\cdot\,;\mathbf{a}) : \times_{i \in \Pi_b} \mathbb{S}_i \times \mathbb{S}^{\Pi}_{(b)}(\mathbf{a}) \times \Omega_r \to \mathbb{R}^{\ell_b} \quad b = 1,...,B,$$

such that

$$r^{\Pi}_{[b]}(Y^{\Pi}_{[b]}(\omega), Z^{\Pi}_{(b)}(\omega_s), \omega_r; \mathbf{a}) = \mathbf{0}, \;\; b = 1,...,B,$$

for each $\omega := (\omega_r,\omega_s) \in \Omega_r \times \Omega^{\Pi}_{(b)}(\mathbf{a})$, $\Omega^{\Pi}_{(b)}(\mathbf{a}) := \{\omega_s : Z^{\Pi}_{(b)}(\omega_s) \in \mathbb{S}^{\Pi}_{(b)}(\mathbf{a})\}$, where $Z^{\Pi}_{(b)}$ is the vector containing $Z^{\Pi}_j, j \notin \Pi_b$ and $Y^{\Pi}_{[b]}$ is the vector containing $Y^{\Pi}_i, i \in \Pi_b$. Write

$$\mathcal{X}^{\Pi}_0(\omega;0) \;:= Y_0(\omega_s), \quad \mathcal{X}^{\Pi}_0(\omega;1) := Z_0(\omega_s),$$

$$\mathcal{X}_i^{\Pi}(\omega;0) \ := Y_i^{\Pi}(\omega), \quad \mathcal{X}_i^{\Pi}(\omega;1) := Z_i^{\Pi}(\omega_s),\ i = 1,...,n,$$

so that the fundamental settable variables $\mathcal{X}_0^{\Pi}$ *and* settable variables $\mathcal{X}_i^{\Pi}$, $i = 1,...,n$ *are mappings such that:*

$$\mathcal{X}_0^{\Pi} : \Omega \times \{0,1\} \to \times_{k=1}^{m} \mathbb{S}_{0,k} \text{ and } \mathcal{X}_i^{\Pi} : \Omega \times \{0,1\} \to \mathbb{S}_i,\ i = 1,...,n.$$

Finally, write

$$\mathcal{X}^{\Pi} := (\mathcal{X}_0^{\Pi}, \mathcal{X}_1^{\Pi},,...,\mathcal{X}_n^{\Pi}).$$

Then $\mathcal{S}^{\Pi} := \{(\mathbf{A},\mathbf{a}),(\Omega,\mathcal{F},P_{\mathbf{a}}),(\Pi,\mathcal{X}^{\Pi})\}$ *is a* stochastic settable system.

A stochastic settable system consists of *units* $i = 1,...,n$ with *unit attributes* a_i belonging to *unit attribute space* A. When $m > 0$, the system has optional *fundamental units* $k = 1,...,m$ with fundamental unit attributes $a_{0,k}$ also belonging to A. The *joint attributes* $\mathbf{a} := (a_{0,1},...,a_{0,m},a_1,...,a_n)$ belong to the *joint attribute space* $\mathbf{A}$. By construction, the unit attributes include the *admissible settings* $\mathbb{S}_i$ for each unit ($\mathbb{S}_{0,k}$ for any fundamental units). $\mathbb{S}_i$ must contain at least two values, necessary to ensuring that interventions are well defined.

The probability space $(\Omega,\mathcal{F},P_{\mathbf{a}})$ embodies the stochastic structure of the system. By representing the *primary settings* as elements $\omega := (\omega_r,\omega_s)$ of $\Omega := \Omega_r \times \Omega_s$, we provide explicit means for variation-free interventions to primary setting values ω_r and the remaining setting values (via ω_s). By "variation-free", we mean that we can consider interventions $(\omega_1,\omega_2) = ((\omega_{r,1},\omega_s),(\omega_{r,2},\omega_s))$ in which only the ω_r component differs or interventions $(\omega_1,\omega_2) = ((\omega_r,\omega_{s,1}),(\omega_r,\omega_{s,2}))$ in which only the ω_s component differs. Requiring that Ω_r and Ω_s each have at least two elements ensures that interventions to ω_r and ω_s are well defined.

The probability measure $P_{\mathbf{a}}$ is indexed by the attributes $\mathbf{a}$ and governs the joint distribution of the random settings and responses. $P_{\mathbf{a}}$ may be determined by nature, determined by a researcher, or determined in part by nature and in part by a researcher. $P_{\mathbf{a}}$ can embody any probabilistically meaningful dependence or independence for events involving ω_r and ω_s. Completeness of the probability space is a technical requirement ensuring that the collection of events $\mathcal{F}$ contains every subset of any event having $P_{\mathbf{a}}-$probability zero.

We call the random variables $Z_i^{\Pi}(\cdot)$ *settings* and realizations $Z_i^{\Pi}(\omega_s)$ *setting values*. By suitably choosing Ω_s and Z_i^{Π}, we also achieve variation-free interventions for the individual setting values. Specifically, let $\Omega_s := (\times_{k=1}^{m}\mathbb{S}_{0,k}) \times (\times_{i=1}^{n}\mathbb{S}_i)$, so that Ω_s has typical element $\omega_s := (z_{0,1},...,z_{0,m}, z_1,...,z_n)$. Further, let $Z_{0,k}(\omega_s) = z_{0,k}$ and $Z_i^{\Pi}(\omega_s) = z_i$ be the projection functions that select the specified component of ω_s. By construction, these functions are surjective (onto). That is, the range $Z_i^{\Pi}(\Omega_s)$ equals the co-domain $\mathbb{S}_i$, so that there is (at least) one ω_s corresponding to each admissible value in $\mathbb{S}_i$. With a suitable choice of $\mathcal{F}_s$ (e.g., that generated by the measurable finite dimensional product cylinders), these choices for $Z_{0,k}$ and Z_i^{Π} are also measurable, as required. (White (2001, Section 3.3) provides relevant background and discussion.) Thus, different values $\omega_{s,1}$ and $\omega_{s,2}$ can generate interventions referencing just a single settable variable, so that $Z_i^{\Pi}(\omega_{s,1}) \neq Z_i^{\Pi}(\omega_{s,2})$, but $Z_j^{\Pi}(\omega_{s,1}) = Z_j^{\Pi}(\omega_{s,2})$ for $j \neq i$. Further, when surjectivity holds, it ensures that the primary interventions $(\omega_{s,1},\omega_{s,2})$ can represent every admissible intervention to the setting values.

When the system has fundamental units, these units have *fundamental responses* $Y_{0,k}$; these are random variables whose values $Y_{0,k}(\omega_s)$ are determined solely by $\omega_s \in \Omega_s$. By convention, *fundamental settings* $Z_{0,k}$ are random variables identical to $Y_{0,k}$. When $Y_{0,k}$ is surjective, then so is $Z_{0,k}$.

Each element Π_b of the partition $\Pi = \{\Pi_b\}$ identifies a group of (non-fundamental) units. The *joint response function* $r^{\Pi}_{[b]}$ specifies how these identified units jointly and freely respond to given jointly admissible setting values of all units not belonging to Π_b.

The given values are setting values $Z^{\Pi}_j(\omega_s)$ for j not belonging to Π_b, including $Z_0(\omega_s)$ and ω_r, represented here by $(Z^{\Pi}_{(b)}(\omega_s), \omega_r)$. The values $Z^{\Pi}_{(b)}(\omega_s)$ belong to the set of *jointly admissible setting values* $\mathbb{S}^{\Pi}_{(b)}(\mathbf{a})$, a subset of $\times_{j\notin\Pi_b}\mathbb{S}_j \times^m_{k=1}\mathbb{S}_{0,k}$. In the absence of constraints, we have $\mathbb{S}^{\Pi}_{(b)}(\mathbf{a}) = \times_{j\notin\Pi_b}\mathbb{S}_j \times^m_{k=1}\mathbb{S}_{0,k}$. Often, however, applications place joint restrictions on the admissible setting values. For example, when the settings represent probabilities (as in the mixed strategy games considered shortly), the constraint that probabilities add to one jointly restricts admissible setting values. The constraints are encoded in $\mathbf{a}$, and implemented by $\mathbb{S}^{\Pi}_{(b)}(\mathbf{a})$.

The *response values* are $Y^{\Pi}_{[b]}(\omega)$, the vector containing unit i's response value $Y^{\Pi}_i(\omega)$ for each i in Π_b, satisfying

$$r^{\Pi}_{[b]}(Y^{\Pi}_{[b]}(\omega), Z^{\Pi}_{(b)}(\omega_s), \omega_r; \mathbf{a}) = \mathbf{0}. \tag{10}$$

Note that we do not explicitly require that $Y^{\Pi}_{[b]}(\omega)$ is the unique solution to the equations in (10). As discussed in our machine learning examples, the governing principles of the system (e.g., optimization and/or equilibrium) operate to deliver a selected system response satisfying these equations. By including the governing principles (including appropriate selection operators) among the attributes $\mathbf{a}$, as is fully rigorous and proper, the presence of $\mathbf{a}$ in the response function can ensure a unique response value. Note that the response function depends on the full system attribute vector $\mathbf{a}$, not just the attributes associated with the units of the given block b. This has been a common feature of our examples. We call the random variables $Y^{\Pi}_i(\cdot)$ *responses*.

Our expression for the responses is in implicit form, as is appropriate for solutions of optimization problems. Nevertheless, it is often convenient to abuse notation somewhat and write response values explicitly as

$$Y^{\Pi}_{[b]}(\omega) = r^{\Pi}_{[b]}(Z^{\Pi}_{(b)}(\omega_s), \omega_r; \mathbf{a}).$$

Because the partition is exhaustive, the collection of response functions $r^{\Pi} := (r^{\Pi}_{[1]}, ..., r^{\Pi}_{[B]})$ provides a description of how each unit in the system responds when it is free to do so in the company of other specified freely responding units. In given circumstances, it may be that only one of these sets of responses is factual; the others are then counterfactual.

Settable variables $\mathcal{X}^{\Pi}_i : \Omega \times \{0,1\} \to \mathbb{S}_i$ embody the dual aspects of settings and responses. Responses $\mathcal{X}^{\Pi}_i(\,\cdot\,,0) := Y^{\Pi}_i$ are random variables taking values in $\mathbb{S}_i$ in response to settings of other settable variables of the system outside the block to which i belongs, say Π_b. The settings $\mathcal{X}^{\Pi}_i(\,\cdot\,,1) := Z^{\Pi}_i$ are random variables taking values in $\mathbb{S}_i$ whose realized values determine the realized responses of other settable variables. The optional fundamental settable variables $\mathcal{X}^{\Pi}_0 : \Omega \times \{0,1\} \to \times^m_{k=1}\mathbb{S}_{0,k}$ yield identical random responses and settings whose values drive responses of other settable variables. We collect together all settable variables of the system by writing $\mathcal{X}^{\Pi} := (\mathcal{X}^{\Pi}_{0,1}, ..., \mathcal{X}^{\Pi}_{0,m}, \mathcal{X}^{\Pi}_1, ..., \mathcal{X}^{\Pi}_n)$. Observe that $\mathcal{X}^{\Pi}$ actually depends on $\mathbf{a}$ through the response functions r^{Π}, so it would be formally correct and more explicit to write $\mathcal{X}^{\Pi}_{\mathbf{a}}$ instead of $\mathcal{X}^{\Pi}$. We forego this for notational simplicity, but this dependence should not be overlooked.

Our notation for the stochastic settable system, $\mathcal{S}^{\Pi} := \{(\mathbf{A}, \mathbf{a}), (\Omega, \mathcal{F}, P_{\mathbf{a}}), (\Pi, \mathcal{X}^{\Pi})\}$, references each component of the system in a way that expresses the hierarchy of these components. At the lowest level is the *attribute structure*, $(\mathbf{A}, \mathbf{a})$; next comes the *stochastic structure*, $(\Omega, \mathcal{F}, P_{\mathbf{a}})$; resting on these is the *causal structure*, $(\Pi, \mathcal{X}^{\Pi})$.

6. Game Theory, Settable Systems, and the PCM

So far, our machine learning examples have shown how settable systems apply to decision problems where optimization operates as a governing principle. We now discuss examples showing how settable systems apply to groups of interacting and strategically competitive decision-making agents. In economics, these agents are usually viewed as consumers, firms, and/or government entities. Of direct relevance to machine learning is that agents may also be artificial intelligences, as in automated trading systems. In addition to their empirical relevance (e.g., the analysis of FCC spectrum auctions or U.S. Treasury Bill auctions), such environments present the opportunity for emergent and distributed computation of otherwise difficult to compute quantities, like prices.

Game theory, the study of multi-agent decision problems, provides a rich formal framework in which to understand and explain the behavior of interacting decision makers. Gibbons (1992) provides an excellent introduction. By showing how the structures of game theory map to settable systems, we establish the foundations for causal analysis of such systems. A central feature of such structures is that their outcomes are determined by suitable equilibrium mechanisms, specifically *Nash equilibrium* and its refinements. Among other things, these mechanisms play a key role in ensuring the mutual consistency of various partitions relevant to the analysis of a given game.

6.1 Pure-Strategy Games and Pure-Strategy Nash Equilibria

The simplest games are static games of complete information (Gibbons, 1992, Chap. 1). In these games, each of n players has: (i) a number of playable strategies (let player i have K_i playable strategies, $s_{i,1},...,s_{i,K_i}$); and (ii) a utility (or "payoff") function u_i that describes the payoff π_i to that player when each player plays one of their given strategies. That is, $\pi_i = u_i(s_1,...,s_n)$, where $s_j \in S_j := \{s_{j,1},...,s_{j,K_j}\}$, $j = 1,...,n$. The players simultaneously choose their strategies; then each receives the payoff specified by the collection of the jointly chosen strategies and the players' payoff functions. Such games are "static" because of the simultaneity of choice. They are "complete information" games because the players' possible strategies and payoff functions are known to all players. (Thus, each player can assess the game from every other player's viewpoint.) An n-player static game of complete information is formally represented in "normal form" as $\mathcal{G} = \{S_1,...,S_n; u_1,...,u_n\}$.

These games map directly and explicitly to the settable system framework. Specifically, the players correspond to units $i = 1,...,n$. The unit attributes a_i include the identity attribute i, the strategies $\mathbb{S}_i = S_i$ available to player i, and the player's utility function $u_i : S_1 \times ... \times S_n \to \mathbb{R}$. When a strategy for player i is set arbitrarily, we denote its value as $z_i \in S_i$; when player i chooses a strategy (a response) we represent its value as $y_i \in S_i$. For concreteness and without loss of generality, we take $S_i := \{1,...,K_i\}$. The players' utility functions implicitly account for the possibility that strategy 1 for player i may represent a different action than that of strategy 1 for player j.

Each player seeks to maximize their payoff given the strategies of the others, so that

$$y_i = r_i^e(z_{(i)};\mathbf{a}) = \arg\max_{z_i \in S_i} u_i(z_1,...,z_n).$$

In economics, this goal-seeking behavior is called "rationality;" an equally fitting (or perhaps superior) term is "intelligence." Thus, game theory analyzes rational or intelligent agents.

Here, we write the responses r_i^e using the superscript e to denote that these response are those for the elementary partition, $\Pi^e := \{\Pi_1^e,...,\Pi_n^e\}$ with $\Pi_i^e = \{i\}$, as each response takes the strategies of all other players as fixed.

For convenience, we assume that for each player there is a unique utility-maximizing response, but just as in our previous machine learning examples, we can generally make a principled selection when the optimal decision is a set. Below, we discuss this further.

In game theory, r_i^e is called a "best-response" function. In settable systems, we refer generically to functions like r_i^e as "response" functions, in part motivated by this usage. Because in game theory the specific game $\mathcal{G}$ under consideration is almost always clear, there is usually no need to explicitly reflect its elements in the players' best response functions. The explicit appearance of players' joint attributes $\mathbf{a}$ (which characterize the game) in the response functions $r_i^e(\,\cdot\,;\mathbf{a})$ emphasizes their role in determining player responses.

Now consider the PCM representation of this game. In the PCM, the attributes become background variables u. The attributes $a_i = (i, S_i, u_i)$ do not map directly to PCM background variables, as S_i is a set and u_i is a function; the PCM requires the background variables to be real numbers. Nevertheless, some simple modifications deliver a conforming representation: we can replace S_i with the integers 1 through K_i and u_i with the vector of values taken by $\pi_i = u_i(s_1,...,s_n)$ as the strategies range over all possible values. We collect these values together across all players and write them as u. Endogenous variables $v = \{s_1,...,s_n\}$ represent player strategies, and the structural functions $f = \{f_1,...,f_n\}$ represent the best response for agent i as $s_i = f_i(s_{(i)}, u), \quad i = 1,...,n$.

The final condition required by the PCM is that there exists a unique fixed point, defined by functions g_i such that $s_i = s_i^* := g_i(u), i = 1,...,n$. When such a unique fixed point exists, it represents a *pure-strategy Nash equilibrium* (Nash, 1950). By definition this satisfies

$$u_i(s_1^*,...,s_i^*,...,s_n^*) \geq u_i(s_1^*,...,s_i,...,s_n^*) \;\; for\ all\ s_i \in S_i, i = 1,...,n.$$

Gibbons (1992, p. 8) provides further discussion of pure-strategy Nash equilibrium.

Just as we saw in Section 3, the PCM faces difficulties arising from its requirement of a unique fixed point. A first difficulty for the PCM is that there are important games for which a pure-strategy Nash equilibrium does not exist; the PCM therefore has nothing to say about such games. A leading example of such games is known as *matching pennies* (Gibbons, 1992, p. 29). In this game, each of two players has a penny that they can choose to display face up (heads) or face down (tails). If the pennies match, player 2 gets both; otherwise player 1 gets both. This game applies to any situation in which one player would like to outguess the other, as in *poker* (bluffing), *baseball* (pitcher vs. hitter), and *battle*.

Given the interest attaching to such games, one would like to have an applicable causal model. This need is met by the settable system framework. Because this framework imposes no fixed point requirement, it applies regardless of the existence of a unique pure-strategy Nash equilibrium. For games with no pure-strategy Nash equilibrium, the response functions $r_i^e(z_{(i)};\mathbf{a})$ of the elementary partition $\Pi^e := \{\{1\},...,\{n\}\}$ readily provide complete information about the best response for all counterfactual strategy combinations of the other players.

If a unique pure-strategy Nash equilibrium exists, it has settable system representation

$$s_i^* = r_i^g(\mathbf{a}), \quad i = 1,...,n,$$

where r_i^g is the response function for the global partition, $\Pi^g := \{\{1,...,n\}\}$. An interesting feature of these response functions is that they depend only on the attributes $\mathbf{a}$; no fundamental or even primary settings appear. Observe also that the Nash equilibrium condition ensures the mutual consistency of the elementary and global partitions.

When there is no pure strategy Nash equilibrium, as in the matching pennies game, there need not exist a valid settable system for the global partition. This provides an interesting example in which we have a well-defined settable system for the elementary partition, but not for the global partition. In contrast, the PCM does not apply at all.

Another difficulty for the PCM is that the unique fixed point requirement prevents it from applying to games with multiple pure-strategy Nash equilibria. An example is the game known as *battle of the sexes* (Gibbons, 1992, p. 11). In this game, two players (Ralph and Alice) are trying to decide on what to do on their next night out: attend a boxing match or attend an opera. Each would rather spend the evening together than apart, but Ralph prefers boxing and Alice prefers the opera. With the payoffs suitably arranged (symmetric), there is a unique best response for each player, given the strategy of the other. Nevertheless, this game has two pure-strategy Nash equilibria: (i) both select boxing; (ii) both select the opera. Thus, the PCM does not apply.

In contrast, the settable system framework does apply, as it does not impose a unique fixed point requirement. The elementary partition describes each agent's unique best response to a given strategy of the other. Further, when multiple Nash equilibria exist, the global partition can yield a well-defined settable system by selecting one of the possible equilibria. As Gibbons (1992, p. 12) notes, "In some games with multiple Nash equilibria one equilibrium stands out as the compelling solution to the game," leading to the development of "conventions" that provide standard means for selecting a unique equilibrium from the available possibilities.

An example is the classic *coordination game*, in which there are two pure-strategy Nash equilibria, but one yields greater payoffs to both players. The convention is to select the higher payoff equilibrium. If such a convention exists, the global partition can specify the response functions r_i^g to deliver this. In such cases, the global partition responses satisfy not only a fixed-point property, but also embody equilibrium selection.

Interestingly, battle of the sexes is not a game with such a convention, as both equilibria seem equally compelling. A more elaborate version of this game, involving incomplete information, does possess a unique equilibrium, however (Gibbons, 1992, pp. 152-154).

6.2 Mixed-Strategy Games and Mixed-Strategy Nash Equilibria

As just suggested, one can modify the character of a game's equilibrium set by elaborating the game. Specifically, consider "mixed-strategy" static games of complete information. Instead of optimally choosing a pure strategy, each player h now chooses a vector of probabilities $p_h := (p_{h,1}, ..., p_{h,K_h})$ (a mixed strategy) over their available pure strategies, say $S_h := \{1, ..., K_h\}$, so that $p_{h,j}$ is the probability that player h plays strategy $j \in S_h$. For example, the probability vector $(1, 0..., 0)$ for player h represents playing the pure strategy $s_h = 1$.

Note that we have modified the notation for the player index from i to h. This enables us to continue to index units using i. Here, the units i correspond to *agent-decision pairs* (h, j). The values h and j become part of the unit attributes, a_i. When referencing i, we may for convenience reference the corresponding h, j, so, for example, we may write a_i or $a_{h,j}$, whichever is more convenient.

Each player h now behaves rationally or intelligently by choosing mixed-strategy probabilities to maximize their expected payoff given other players' strategies,

$$\bar{\pi}_h = \upsilon_h(p^n) := \sum_{s^n \in S^n} u_h(s^n) \Pr(s^n; p^n),$$

where for conciseness we now write $s^n := (s_1, ..., s_n)$, $S^n := S_1 \times ... \times S_n$, and $p^n := (p_1, ..., p_n)$. (Maximizing expected payoff is not the only possibility, but we focus on this case for concreteness.) The strategies are chosen independently, so that $\Pr(s^n; p^n)$, the probability that the agents jointly choose the configuration of strategies s^n, is given by $\Pr(s^n; p^n) = \prod_{h=1}^{n} p_{h,s_h}$.

It is a famous theorem of Nash (1950) that if n is finite and if K_h is finite, $h = 1, ..., n$, then there must exist at least one Nash equilibrium for $\mathcal{G}$, possibly involving mixed strategies (e.g., Gibbons, 1992, p. 45).

We map mixed-strategy games to settable systems as follows. As mentioned above, units i are agent-decision pairs (h, j), so that unit attributes a_i include the agent and decision designators, h and j. Because settings and responses are now probabilities, unit attributes also specify admissible settings $\mathbb{S}_{h,j}$ as a subset of $[0, 1]$. We further discuss $a_{h,j}$ below.

For each agent h, there is a $K_h \times 1$ vector of settings and responses. We denote the probabilities of the mixed strategy for agent h as $z_{h,j}, j = 1, ..., K_h$, when these are set, and as $y_{h,j}, j = 1, ..., K_h$, when these constitute the agent's best response. Let z_h be the $K_h \times 1$ vector with elements $z_{h,j}$, and let y_h be the $K_h \times 1$ vector with elements $y_{h,j}$. Given all other player's mixed strategies $z_{(h)}$, agent h's best response is

$$y_h = r_h^a(z_{(h)}; \mathbf{a}) = \sigma_h(\arg\max_{z_h \in \mathbf{S}_h} \upsilon_h(z_1, ..., z_n)),$$

where the maximization is taken over the simplex $\mathbf{S}_h := \{z \in [0, 1]^{K_h} : \sum_{j=1}^{K_h} z_j = 1\}$. The operator σ_h performs a measurable selection, discussed below.

Several aspects of this representation are notable. First, we write the response function r_h^a to denote that it is the response function for the *agent partition* $\Pi^a := \{\Pi_h^a, h = 1, ..., n\}$, where $\Pi_h^a = \{(h, 1), ..., (h, K_h)\}$. In contrast, the elementary partition is $\Pi^e := \{\Pi_{h,j}^e, j = 1, ..., K_h; h = 1, ..., n\}$, with $\Pi_{h,j}^e := \{(h, j)\}$. The response functions $r_{h,j}^e$ for the elementary partition describe the best response for agent h's strategy j given not only all other agents' strategies, but also all other strategies for agent h. The elementary partition is usually not of particular interest; the agent partition and the global partition are typically the main objects of interest in this context.

The superscript ^{a} in r_h^a and elsewhere to denote the agent partition creates no confusion with the joint attributes $\mathbf{a}$, as the former is always a superscript, and the latter never is.

Next, we note that the unit attributes $a_{h,j}$ contain admissible values $\mathbb{S}_{h,j} \subset [0, 1]$, so that $0 \leq z_{h,j} \leq 1$. This is not enough to fully specify the admissible values for the vector z_h, however, as the probabilities must add up to 1. This means that z_h must belong to the simplex $\mathbf{S}_h$. We enforce this constraint by making $\mathbf{S}_h$ a component of each unit attribute $a_{h,j}, j = 1, ..., K_h$. Just as an attribute common to all system units is a system attribute, any attribute common to a given subset of units is an attribute of that subset. Thus, $\mathbf{S}_h$ is an attribute of agent h; agent h is that subset of the units with agent designator equal to h.

An interesting feature of mixed-strategy games is that the set $\arg\max_{z_h \in \mathbf{S}_h} \upsilon_h(z_1, ..., z_n)$ can easily fail to have a unique element. This set thus defines the player's best-response correspondence, rather than simply giving a best-response function. We obtain a best-response function by applying a measurable selection operator σ_h to the set of maximizers. The operator σ_h is an attribute, specifically of agent h; thus, we include it as a component of the unit attributes $a_{h,j}, j = 1, ..., K_h$.

By definition, the agent is indifferent between elements of the arg-max set; the choice of selection operator is not crucial. In fact, the selection may be random, implemented by letting σ_h depend

on $\omega_r \in \Omega_r$, so that one has

$$y_h = r_h^a(z_{(h)}, \omega_r; \mathbf{a}) = \sigma_h(\arg\max_{z_h \in \mathbf{S}_h} \upsilon_h(z_1, ..., z_n), \omega_r).$$

Now consider how this game maps to the PCM. Again, the attributes map to the background variables u, although now the attributes are, among other things, sets with a continuum of values and correspondences. Mapping these to a vector of real numbers is problematic, so we simply view u as a general vector whose elements may be numbers, sets, functions, or correspondences. The endogenous variables are most appropriately represented as $K_h \times 1$ vectors p_h such that $v = \{p_1, ..., p_n\}$. The elements of $f := \{f_1, ..., f_n\}$ are correspondingly vector-valued. These must satisfy $p_h = f_h(p_{(h)}, u) := \sigma_h(\arg\max_{p_h \in \mathbf{S}_h} \upsilon_h(p_1, ..., p_n))$.

In order to apply the PCM, we require a unique fixed point. Even when a unique Nash equilibrium exists, to obtain this as the fixed point requires choosing the selection operators σ_h so that they specifically produce the Nash equilibrium response. In the usual situation, the properties of f determine whether or not a fixed point exists. Here, however, knowledge of the unique fixed point is required to properly specify σ_h, hence f_h, an awkward reversal signaling that the PCM is not well-suited to this application. Indeed, the selection cannot be random, a plausible response when the player is indifferent between different strategies.

An interesting feature of this example is that when the PCM applies, it does so with vector-valued units rather than the scalar-valued units formally treated by Pearl (2000) or Halpern (2000). The PCM is thus necessarily silent about what happens when components of an agent's strategy are arbitrarily set. In contrast, settable systems apply to partitions both finer and coarser than the agent partition. (The elements (sets of unit indexes) of a "coarser" partition are unions of the elements of a "finer" partition. Thus, the agent partition is coarser than the elementary partition and finer than the global partition.)

Unlike the case of pure-strategy games, there must always be at least one mixed-strategy Nash equilibrium, so the PCM does not run into the difficulty that there may be no equilibrium. Nevertheless, mixed-strategy games can also have multiple Nash equilibria, so the PCM does not apply there. For a given game, the GPCM does apply to the agent partition, but it does not incorporate equilibrium selection mechanisms. In contrast, the settable system framework permits causal analysis at the level of the agent partition (as well as coarser or finer partitions); represents the unique Nash equilibrium at the level of the global partition without requiring a selection operator when a unique equilibrium exists; and otherwise represents the desired responses when a unique mixed-strategy Nash equilibrium does not exist but conventions or other plausible selection mechanisms apply.

Static games of complete information are the beginning of a sequence of increasingly richer games, including dynamic games of complete information, static games of incomplete information, and dynamic games of incomplete information. Each of these games employs progressively stronger equilibrium concepts that rule out implausible equilibria that would survive under equilibrium concepts suitable for simpler games (Gibbons, 1992, p. 173). These implausible equilibria all satisfy fixed-point (simple Nash equilibrium) requirements.

The unique fixed point requirement of the PCM thus acts to severely limit its applicability in game theory, due to the many opportunities for multiple Nash equilibria. Although GPCMs formally apply, they cannot support discourse about causal relations between endogenous variables, due to the lack of an analog of the potential response function. In contrast, by exploiting attributes and partitioning, settable systems permit implementation of whichever stronger and/or more re-

fined equilibria criteria are natural for a given game, together with any natural equilibrium selection mechanism.

6.3 Infinitely Repeated Dynamic Games

Dynamic games are played sequentially. For example, two players can repeatedly play *prisoner's dilemma*. In infinitely repeated games, play proceeds indefinitely. Clearly, infinite repetition cannot be handled in a finite system, so the PCM cannot apply.

In infinitely repeated dynamic games of complete and perfect information (see Gibbons, 1992, Section 2.3.B), players play a given static game in "stages" or periods $t = 1, 2, ...$ The period t payoff to player h is $\pi_{h,t} = u_{h,t}(\alpha_{1,t}, ..., \alpha_{n,t})$, where $u_{h,t}$ is player h's payoff function for period t, whose arguments are the "actions" $\alpha_{j,t}$ at time t of all n players. (The strategies of static games correspond to the actions of dynamic games.) Information is "complete," as each player knows the others' possible actions and payoff functions. Information is "perfect," as at every t, each player knows the entire history of play up to that period.

Rational players act to maximize their average present discounted value of payoff,

$$\bar{\pi}_h = \bar{u}_h(\alpha_1, ..., \alpha_n) := (1-\delta)\sum_{t=1}^{\infty} \delta^{t-1} u_{h,t}(\alpha_{1,t}, ..., \alpha_{n,t}),$$

where $\alpha_j := \{\alpha_{j,t}\}$ denotes player j's countable sequence of actions, and $0 < \delta < 1$ is a "discount rate" (common across players, for simplicity) that converts payoffs $\pi_{h,t}$ in period t to a value in period 1 as $\delta^{t-1}\pi_{h,t}$. A player's best response to any collective sequence of actions by the others is a solution to the problem

$$\max_{s_h \in S_h} \bar{u}_h(\alpha_1, ..., \alpha_n) \;\; subject\ to \;\; \alpha_{h,t} = s_{h,t}(\alpha_1^{t-1}, ..., \alpha_n^{t-1}), \;\; t = 1, 2, ...,$$

where S_h is player h's set of all admissible sequences $s_h := \{s_{h,t}\}$ of "strategy functions" $s_{h,t}$. These represent player h's action in period t as a function only of the prior histories of player actions, $\alpha_1^{t-1}, ..., \alpha_n^{t-1}$. (For $t = 1$, $s_{h,t}$ is a constant function.) Player h's best responses are $\alpha_{h,t}^* = s_{h,t}^*(\alpha_1^{t-1}, ..., \alpha_h^{*t-1}, ..., \alpha_n^{t-1})$, $t = 1, 2, ...$, where $s_h^* := \{s_{h,t}^*\}$ is a sequence of best response strategy functions. (These need not be unique, so $s_{h,t}^*$ may be a correspondence. The player is indifferent among the different possibilities.)

Such games generally have multiple Nash equilibria. Many of these are implausible, however, as they involve non-credible threats; and credibility is central to all dynamic games (see Gibbons, 1992, p. 55). Non-credible equilibria can be eliminated by retaining only "subgame perfect" Nash equilibria. These are Nash equilibria that solve not only the game beginning at time 1, but also the same game beginning at any time $t > 1$ (Gibbons, 1992, pp. 94-95). A celebrated result by James Friedman (1971) ensures the existence of one or more subgame perfect Nash equilibria, provided δ is close enough to one (see, e.g.. Gibbons, 1992, pp. 97-102). Significantly, such equilibria permit *tacit cooperation*, yielding outcomes superior to what players can achieve in the static game played at each stage.

We now map this game to settable systems. The units i now correspond to *agent-time pairs* (h,t). As $t = 1, 2, ...$, there is a countable infinity of units. Agent attributes include their admissible actions and their payoff functions for each period. That is, a_i (equivalently $a_{h,t}$) includes the admissible sequence of functions S_h, the utility function $u_{h,t}$, and the discount factor δ. When player actions

$\alpha_{h,t}$ are set arbitrarily, the settable system represents them as $z_{h,t}$. When players intelligently choose their actions, they are denoted $y_{h,t}$.

The agent partition $\Pi^a := \{\Pi^a_h, h = 1,...,n\}$, where $\Pi^a_h := \{(h,1),(h,2),...\}$, represents agents' best responses recursively as

$$y_{h,t} = \sigma_{h,t}(s^*_{h,t}(z_1^{t-1},...,y_h^{t-1},...,z_n^{t-1}),\omega_r), \;\; t = 1,2,...;h = 1,...,n,$$

where $\sigma_{h,t}$ is a measurable selection operator; $s^*_{h,t}$ is the agent's best response correspondence, which depends on the action histories of other agents, $z^{t-1}_{(h)}$, and agent h's history of prior best responses, y_h^{t-1}; and the realization ω_r determines random selections from the best response correspondence for agent h in period t. Recursive substitution for the elements of the history y_h^{t-1} yields a representation in terms of an agent-partition response function, $r^a_{h,t}$, namely

$$y_{h,t} = r^a_{h,t}(z^{t-1}_{(h)},\omega_r;\mathbf{a}), \;\; t = 1,2,...;h = 1,...,n.$$

The global partition represents whatever selection of the collection of subgame perfect Nash equilibria is natural or compelling. Equilibrium agent responses are given by the global-partition response functions $r^g_{h,t}$ as

$$y_{h,t} = r^g_{h,t}(\omega_r;\mathbf{a}) \quad t = 1,2,...;h = 1,...,n.$$

Notably, this example exploits each feature of stochastic settable systems, including countable dimension, attributes, partitioning, and pure randomness.

6.4 Settable Systems and Multi-Agent Influence Diagrams

Causal models other than the PCM are available in the machine learning literature. We focus here on the PCM because of its prevalence and to maintain a sharp focus for this paper.

A framework particularly related to the preceding discussion is that of Koller and Milch (2003) (KM), who introduce multi-agent influence diagrams (MAIDs) to represent noncooperative games. In particular, KM provide a graphical criterion for determining a notion of "strategic relevance." KM's "relevance graphs" are related to causal graphs. By casting games in the settable system framework, we can immediately construct causal graphs for games by applying the conventions of Section 3.6.2.

The most immediate similarity between settable systems and MAIDs is that they are both capable of modeling environments in which multiple agents interact. In contrast, "influence diagrams [...] have been investigated almost entirely in a single-agent setting" (KM, 2003, p. 189-190). Nevertheless, several features of settable systems distinguish them from MAIDs:

(i) A settable system is an explicit causal framework in which notions of partitioning, settings, interventions, responses, and causality are formally defined. Furthermore, the interrelations between these causal notions on the one hand and the notions of optimization, equilibrium, and learning on the other are made precise. In contrast, there is no formal mention of causality in KM.

(ii) MAIDs build on the "chain rule for Bayesian Networks" (KM, definition 2.2, p. 186). This is equivalent to assuming a set of (conditional) independence relations involving chance and decision variables and is necessary for the applicability of the "s-reachability" graphical criterion. On the other hand, settable systems permit but do not require any assumptions on the joint distribution of

settings and responses. In particular, responses may admit an aspect of "pure randomness" due to their direct dependence on the primary variable.

(*iii*) In KM, an agent's utility is additive (KM, p. 189-190). Settable systems do not impose this requirement.

(*iv*) The KM algorithm for finding Nash equilibria outputs one Nash equilibrium. It selects an equilibrium arbitrarily if multiple equilibria are found. Further, the algorithm cannot produce certain equilibria, such as a nonsubgame-perfect equilibrium (KM, p. 216). The settable system framework can represent principled selections from all relevant Nash equilibria.

We emphasize that the results in KM are very helpful for representing and studying games. In particular, under the MAID assumptions, the KM results permit an explicit representation of games and can lead to computational savings.

7. Machine Learning, Optimization, and Equilibrium

A general learning algorithm introduced by Kushner and Clark (1978) (KC) has the form

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \lambda_t M_t(\hat{\xi}_t, \hat{\theta}_t, \zeta_{t+1}), \tag{11}$$

$$\hat{\xi}_{t+1} = R_t(\hat{\xi}^t, \hat{\theta}^{t+1}, \zeta_{t+1}), \quad t = 0, 1, 2, ..., \tag{12}$$

where $\hat{\theta}_t$ and $\hat{\xi}_t$ are random vectors, λ_t is a random scalar, M_t and R_t are known vector functions, $\hat{\xi}^t := (\hat{\xi}_0, ..., \hat{\xi}_t)$, $\hat{\theta}^{t+1} := (\hat{\theta}_0, ..., \hat{\theta}_{t+1})$, and ζ_t is an observable random vector. Initial values $\hat{\xi}_0$ and $\hat{\theta}_0$ are random vectors independent of $\{\zeta_t\}$. KC call this a Robbins and Monro (1951) (RM) algorithm with feedback (RMF). Equation (11) is an RM procedure; Equation (12) supplies the feedback. A main focus of interest is the convergence behavior of $\hat{\theta}_t$ as $t \to \infty$.

Chen and White (1998) analyze a version of RMF where each vector takes values in a real separable infinite-dimensional Hilbert space. We call this an HRMF algorithm. Because of the flexibility this affords, the HRMF supports nonparametric learning.

The RM procedure emerges when $\hat{\xi}_t$ has dimension zero, so $\hat{\theta}_{t+1} = \hat{\theta}_t + \lambda_t M_t(\hat{\theta}_t, \zeta_{t+1})$, $t = 0, 1, 2, ...$ This contains recursive least squares (e.g., back-propagation), recursive maximum likelihood, and recursive method of moments procedures (e.g., Ljung and Soderstrom, 1983). The estimated weights are $\hat{\theta}_t$; $\{\zeta_t\}$ is the data sequence; λ_t is the "learning rate," for example, $\lambda_t = 1/t$; and M_t determines the learning method (least squares, maximum likelihood, etc.). By permitting feedback, the RMF accommodates the evolution of internal, possibly hidden states $\hat{\xi}_t$; thus, Kalman filter methods (Kalman, 1960) are a special case.

The RMF also describes learning in recurrent artificial neural networks (ANNs) (e.g., Elman, 1990; Jordan, 1992; Kuan, Hornik, and White, 1994). Here, the input sequence is $\{\zeta_t\}$; after t input observations, network weights are $\hat{\theta}_t$, and hidden unit activations are $\hat{\xi}_t$. The learning function is M_t, the learning rate is λ_t, and R_t determines hidden unit activations. The allowed randomness of λ_t accommodates simulated annealing.

The RMF and HRMF contain systems with learning by one or more optimizing agents. When there are multiple agents, the system can embody convergence to equilibrium. Specifically, Chen and White (1998) provide conditions ensuring system convergence as $t \to \infty$ to Nash equilibria or to "rational expectations" equilibria. As examples, Chen and White (1998) consider, among others, a learning agent solving a *stochastic dynamic programming* problem and the game of *fictitious play with continuum strategies* (an infinitely repeated dynamic game of incomplete information). The applications of the (H)RMF are thus broad; further, the settable systems framework contains both.

The units i are *time-agent-generalized decision triples*, (t,h,j). Specifically, at time t, agent h has generalized decisions indexed by j. Generalized decisions are *knowledge* (or in Bayesian frameworks, *beliefs*) denoted $\hat{\theta}_{t,h,k}$, $k = 1,...,k_h$, or *generalized actions*, denoted $\hat{\xi}_{t,h,\ell}$, $\ell = 1,...,\ell_h$. Generalized actions may be *actions* (as in Section 6.3) or *states*, as in the Kalman filtering and recurrent ANN examples. We write $\hat{\theta}_t := (\hat{\theta}'_{t,1},...,\hat{\theta}'_{t,n})'$, where $\hat{\theta}_{t,h} := (\hat{\theta}_{t,h,1},...,\hat{\theta}_{t,h,k_h})'$ takes values in Θ_h, a subset of $\mathbb{R}^{k_h}$, and $\hat{\xi}_t := (\hat{\xi}'_{t,1},...,\hat{\xi}'_{t,n})'$, where $\hat{\xi}_{t,h} := (\hat{\xi}_{t,h,1},...,\hat{\xi}_{t,h,\ell_h})'$ takes values in Ξ_h, a subset of $\mathbb{R}^{\ell_h}$, $h = 1,...,n$.

In addition to the time-agent-generalized decision indicators (t,h,j), attributes $a_{t,h,j}$ include as components the spaces Θ_h and Ξ_h and the function $\lambda_t : \Omega_r \to \mathbb{R}$. They can also include the functions $M_{t,h,k}$ or $R_{t,h,\ell}$, as appropriate. We write $M_t := (M'_{t,1},...,M'_{t,n})'$, with $M_{t,h} := (M_{t,h,1},...,M_{t,h,k_h})'$, and $R_t := (R'_{t,1},...,R'_{t,n})'$, with $R_{t,h} := (R_{t,h,1},...,R_{t,h,\ell_h})'$. The functions $M_{t,h,k}$ may be a consequence of an underlying optimization principle, as in our machine learning examples of Section 3. The same may be true of the functions $R_{t,h,\ell}$.

For the RMF, in Equations (11) and (12), n is finite, as are k_h and ℓ_h. Because t takes a countable infinity of values, we require a countably infinite settable system. For the HRMF, n may be countably infinite; similarly, k_h and/or ℓ_h may be countably infinite.

Equations (11) and (12) form a recursive or acyclic system. In such systems, there is a natural hierarchy of units, in which *predecessor* units drive *successor* units. The system evolves naturally (i.e., without intervention) when the response values at a given level of the hierarchy act as setting values for successors. Stochastic settable systems are sufficiently flexible to permit this. That is, given recursivity, for every ω_r in Ω_r, there exists ω_s in Ω_s such that $Z_i(\omega_s) = Y_i(\omega_r,\omega_s)$ for all i. When $\omega = (\omega_r,\omega_s)$ has this property, we call ω *canonical*, and we let $\Omega_c \subset \Omega$ denote the set of canonical primary settings ω. Response and setting values for a given unit thus coincide on Ω_c, implementing the natural evolution. In the present example, $Y_{t,h,j}$ and $Z_{t,h,j}$ correspond to an element of either $\hat{\theta}_t$ or $\hat{\xi}_t$. Fundamental settings are $\hat{\xi}_0, \hat{\theta}_0$, and $\{\zeta_t\}$, corresponding to elements of $\mathcal{X}_0(\,\cdot\,,1) := \mathcal{X}_0(\,\cdot\,,0)$.

Substituting Equation (11) into Equation (12) yields response functions for the *time partition*, $\Pi^t := \{\Pi^t_1,\Pi^t_2,...\}$, where $\Pi^t_b := \{(b,h,k), k = 1,...,k_h; (b,h,\ell), \ell = 1,...,\ell_h; h = 1,...,n\}$.

In the HRMF, agents' generalized decisions take values in real separable infinite-dimensional Hilbert spaces, so generalized decisions are not just vector-valued; their values may be suitably well-behaved functions. First, consider how a countably dimensioned settable system accommodates such objects when there is a single agent with a single action, a function, and a single knowledge element, also a function. We represent such functions by a countable vector whose elements are coefficients of terms in a suitable series representation, for example, a Fourier series. Further, this same approach applies without exhausting the dimensionality of the settable system, even when there is a countable infinity of agents, each having a countable infinity of knowledge elements and actions, which are themselves elements of real separable infinite-dimensional Hilbert spaces.

8. Summary and Concluding Remarks

This paper introduces settable systems, an extension of Pearl's (2000) causal model. Settable systems and the PCM share many common features. For example, in both frameworks the variables of the system have a dual role (set or free), there are mechanisms for specifying which variables are set or free (submodels and the do operator in the PCM, partitioning in settable systems), and attributes

may be accommodated (as background variables in the PCM and as a priori constants in settable systems).

The key difference between the PCM and settable systems is the way these common features interrelate to one another. Although we point out a number of limitations of the PCM in motivating settable systems, settable systems strictly build on the percepts of the PCM. Our intent is to show how modest reconfiguration and refinement of the elements of the PCM considerably enhance its explanatory power.

As we demonstrate, the PCM encounters obstacles when we attempt to apply it to certain machine learning examples. These limitations motivate particular features of settable systems. For example, the unique fixed point requirement of the PCM is a significant limitation. Like Halpern's (2000) GPCM, settable systems do not require the existence of a unique fixed point. The structure of settable systems nevertheless leads to natural notions of counterfactuals, interventions, direct causes, direct effects, and total effects. In contrast, the absence of the potential response function in the GPCM precludes causal discourse.

Another appealing feature of settable systems relative to the PCM is its ability to provide a causal role for structurally exogenous variables. This capability arises because settable systems distinguish between attributes and fundamental settings. In contrast, the PCM lumps together attributes and background variables, so neither can play a causal role. The PCM is silent on whether to treat variables as exogenous or endogenous and on how to specify attributes. In settable systems, the governing principles (e.g., optimization and equilibrium) provide explicit guidance for distinguishing exogenous variables and endogenous variables. Attributes are unambiguously defined as constants (numbers, functions, sets, etc.) associated with system units that define fundamental aspects of the decision problem represented by the settable system.

Our examples in game theory (Section 6) and machine learning with feedback (Section 7) further show that settable systems apply directly to systems where learning and/or optimizing agents interact in a process where outcomes satisfy or converge to appropriate equilibria. Settable systems thus provide rigorous foundations for causal analysis in these empirically relevant and computationally important systems.

These foundations are only a first step in analyzing phenomena conforming to settable systems. A particularly important research area is the study of general primitive conditions ensuring the identification of specific causal effects of interest under varying assumptions about the observability of causes of interest and other ancillary causes and under particular patterns of causal relation. In this context, identification means the equality of causally meaningful objects (e.g., expected effects) with corresponding stochastically meaningful objects, that is, quantities expressible solely as a functional of the joint distribution of observable random variables. When identification holds, it becomes possible to estimate various causal effects from data. Recent work of White (2006), White and Chalak (2007), Schennach et al. (2008), Chalak and White (2008a), and White and Kennedy (2009) provides results for identification and estimation of causal effects under varying assumptions.

Key to ensuring identification of effects of interest in all of these studies are specific independence or conditional independence conditions, for example, the conditional independence of causes of interest from unobservable ancillary causes given other observable variables (covariates). Chalak and White (2008b) provide primitive conditions on recursive settable system structures (in particular the response functions) that either ensure or rule out such independence or conditional independence relations. In pursuing this goal, notions of indirect and total effects of non-primary causes

emerge naturally and play key roles. These results also have direct implications for d-separation and D-separation (e.g., Geiger, Verma, and Pearl, 1990; Pearl, 2000, pp. 16-17).

These studies by no means exhaust the opportunities for deeper understanding and application of settable systems. For example, all of the studies of identification and estimation just mentioned are for recursive structures. Obtaining analogous results for non-recursive structures is of particular interest.

At the outset, we offered this paper as part of a cross-disciplinary dialog between the economics/econometrics community and the machine learning community, with the hope that both communities might gain thereby. For economists, the benefits are clear and precise notions of causal effects that apply broadly to economic structures and, in particular, to the powerful structures of game theory. These causal notions draw heavily on concepts at the heart of the PCM, but surmount a number of limitations that may have held back economists' acceptance of the PCM. For those in the machine learning community, one benefit is the extension of causal notions to systems fundamentally involving optimization, equilibrium, and learning, features common to a broad range of application domains relevant to machine learning. We also hope that the machine learning community, which has so far paid only limited attention to game theory, may begin to consider the possibilities it offers for understanding empirical phenomena and for distributed and emergent computation.

Acknowledgments

The authors are grateful for discussion and comments on previous work by Judea Pearl that stimulated this work, and for the comments and suggestions of James Heckman, Philip Neary, Douglas R. White, Scott White, the editor, and five referees. Any errors are solely the author's responsibility. The present paper is an expanded version of the foundational and definitional material contained in our earlier unpublished paper "A Unified Framework for Defining and Identifying Causal Effects."

Appendix A.

For completeness, we provide a formal definition of a non-stochastic settable system.

Definition 2 (Nonstochastic Settable System) *Let $n \in \bar{\mathbb{N}}^+$, and let A be a non-empty set. For each $i = 1,...,n$, let a_i be a fixed element of A, such that a_i includes a component $\mathbb{S}_i$, a multi-element Borel-measurable subset of $\mathbb{R}$.*

Let $m \in \bar{\mathbb{N}}$. For each $k = 1,...,m$, let $a_{0,k}$ be a fixed element of A, such that $a_{0,k}$ includes a component $\mathbb{S}_{0,k}$, a multi-element Borel-measurable subset of $\mathbb{R}$. Write $a_0 := (a_{0,1},...,a_{0,m})$ and $\mathbf{a} := (a_0, a_1, ..., a_n) \in \mathbf{A} := (\times_{k=1}^{m} A) \times (\times_{i=1}^{n} A)$.

For each $k = 1,...,m$, let $z_{0,k} \in \mathbb{S}_{0,k}$, and put $y_{0,k} := z_{0,k}$. Write $z_0 := (z_{0,1},...,z_{0,m})$ and $y_0 := z_0$.

Let $\Pi = \{\Pi_b\}$ be a partition of $\{1,...,n\}$, with $B := \#\Pi \in \bar{\mathbb{N}}^+$, let $\ell_b := \#\Pi_b$, and let $\mathbf{a}$ determine the multi-element Borel measurable set $\mathbb{S}^{\Pi}_{(b)}(\mathbf{a}) \subset \times_{j \notin \Pi_b} \mathbb{S}_j \times_{k=1}^{m} \mathbb{S}_{0,k}$, $b = 1,...,B$. Suppose there exist measurable functions

$$r^{\Pi}_{[b]}(\,\cdot\,;\mathbf{a}) : \times_{i \in \Pi_b} \mathbb{S}_i \times \mathbb{S}^{\Pi}_{(b)}(\mathbf{a}) \to \mathbb{R}^{\ell_b} \quad b = 1,...,B,$$

and real vectors $y^{\Pi}_{[b]} \in \times_{i \in \Pi_b} \mathbb{S}_i$ *such that for each* $z^{\Pi}_{(b)} \in \mathbb{S}^{\Pi}_{(b)}(\mathbf{a})$,

$$r^{\Pi}_{[b]}(y^{\Pi}_{[b]}, z^{\Pi}_{(b)}; \mathbf{a}) = \mathbf{0}, \;\; b = 1, ..., B.$$

Write $\mathcal{X}^{\Pi}_0(0) := y_0$, $\mathcal{X}^{\Pi}_0(1) := z_0$, $\mathcal{X}^{\Pi}_i(0) := y^{\Pi}_i$, $\mathcal{X}^{\Pi}_i(1) := z^{\Pi}_i$, $i = 1, ..., n$, *so that* $\mathcal{X}^{\Pi}_0 : \{0,1\} \to \times^m_{k=1} \mathbb{S}_{0,k}$ *and* $\mathcal{X}^{\Pi}_i : \{0,1\} \to \mathbb{S}_i, i = 1, ..., n$. *Finally, write*

$$\mathcal{X}^{\Pi} := (\mathcal{X}^{\Pi}_0, \mathcal{X}^{\Pi}_1, , ..., \mathcal{X}^{\Pi}_n).$$

Then $\mathcal{S}^{\Pi} := \{(\mathbf{A}, \mathbf{a}), (\Pi, \mathcal{X}^{\Pi})\}$ *is a* nonstochastic settable system.

References

D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

C. Berge. *Espaces Topologiques*. Paris: Dunod (translation by E.M. Patterson, Topological Spaces. Edinburgh: Oliver and Boyd), 1963.

K. Chalak and H. White. An extended class of instrumental variables for the estimation of causal effects. Technical report, Department of Economics, University of California, San Diego, 2008a.

K. Chalak and H. White. Independence and conditional independence in causal systems. Technical report, Department of Economics, University of California, San Diego, 2008b.

X. Chen and H. White. Nonparametric adaptive learning with feedback. *Journal of Economic Theory*, 82:190–222, 1998.

A.P. Dawid. Influence diagrams for causal modeling and inference. *International Statistical Review*, 70:161–189, 2002.

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

M. Eichler and V. Didelez. Causal reasoning in graphical time-series models. In *Proceedings of the 23rd Annual Conference on Uncertainty in Artificial Intelligence*, 2007.

J. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

F. Fisher. A correspondence principle for simultaneous equations. *Econometrica*, 38:73–92, 1970.

J. Friedman. A noncooperative equilibrium for supergames. *Review of Economic Studies*, 38:1–12, 1971.

D. Geiger, T. S. Verma, and J. Pearl. Identifying independence in Bayesian networks. *Networks*, 20:507–534, 1990.

R. Gibbons. *Game Theory for Applied Economists*. Princeton: Princeton University Press, 1992.

J. Halpern. Axiomatizing causal reasoning. *Journal of Artificial Intelligence Research*, 12:317–337, 2000.

D. Heckerman and R. Shachter. Decision-theoretic foundations for causal reasoning. *Journal of Artificial Intelligence Research*, 3:405–430, 1995.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

G. E. Hinton and T. J. Sejnowski. Learning and relearning in Boltzmann machines. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 282–317. Cambridge: MIT Press, 1986.

G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

P. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81: 945–970, 1986.

R.A. Howard and J. E. Matheson. Influence diagrams. In R.A. Howard and J.E. Matheson, editors, *Readings in the Principles and Applications of Decision Analysis*. Menlo Park, CA: Strategic Decisions Group, 1984.

M. Jordan. Constrained supervised learning. *Journal of Mathematical Psychology*, 36:396–425, 1992.

R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Series D, Journal of Basic Engineering*, 82:35 – 45, 1960.

D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45:181–221, 2003.

C.-M. Kuan, K. Hornik, and H. White. A convergence result for learning in recurrent neural networks. *Neural Networks*, 6:420–440, 1994.

H. Kushner and D. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Berlin: Springer-Verlag, 1978.

L. Ljung and T. Soderstrom. *Theory and Practice of Recursive Identification*. Cambridge, MA: MIT Press, 1983.

J. Marsden and A. Tromba. *Vector Calculus*. New York: W.H. Freeman, 5th edition, 2003.

J. Nash. Equilibrium points in *n*-person games. In *Proceedings of the National Academy of Sciences*, volume 36, pages 48–49, 1950.

L.G. Neuberg. Review of *Causality: Models, Reasoning, and Inference*. *Econometric Theory*, 19: 675–685, 2003.

J. Pearl. *Causality*. New York: Cambridge University Press, 2000.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

S. Schennach, H. White, and K. Chalak. Estimating average marginal effects in nonseparable structural systems. Technical report, Department of Economics, University of California, San Diego, 2008.

Y. Sergeyev and V. Grishagin. Parallel asynchronous global search and the nested optimization scheme. *Journal of Computational Analysis and Applications*, 3:123–145, 2001.

B. Shipley. *Cause and Correlation in Biology: A User's Guide to Path Analysis, Structural Equations and Causal Inference*. Cambridge University Press, 2000.

R. Strotz and H. Wold. Recursive vs. nonrecursive systems: An attempt at synthesis. *Econometrica*, 28:417–427, 1960.

S. van Huffel and P. Lemmerling. *Total Least Squares and Errors-In-Variables Modeling: Analysis, Algorithms and Applications*. Berlin: Springer-Verlag, 2002.

H. White. *Asymptotic Theory for Econometricians*. New York: Academic Press, 2001.

H. White. Time-series estimation of the effects of natural experiments. *Journal of Econometrics*, 135:527–566, 2006.

H. White and K. Chalak. Identifying effects of endogenous causes in nonseparable systems using covariates. Technical report, Department of Economics, University of California, San Diego, 2007.

H. White and P. Kennedy. Retrospective estimation of causal effects through time. In J. Castle and N. Shephard, editors, *The Methodology and Practice of Econometrics: A Festschrift in Honour of David Hendry*. Oxford: Oxford University Press, 2009.

Journal of Machine Learning Research 10 (2009) 1801-1828 Submitted 6/08; Revised 2/09; Published 8/09

Distributed Algorithms for Topic Models

David Newman NEWMAN@UCI.EDU
Arthur Asuncion ASUNCION@ICS.UCI.EDU
Padhraic Smyth SMYTH@ICS.UCI.EDU
Max Welling WELLING@ICS.UCI.EDU
Department of Computer Science
University of California, Irvine
Irvine, CA 92697, USA

Editor: Andrew McCallum

Abstract

We describe distributed algorithms for two widely-used topic models, namely the Latent Dirichlet Allocation (LDA) model, and the Hierarchical Dirichet Process (HDP) model. In our distributed algorithms the data is partitioned across separate processors and inference is done in a parallel, distributed fashion. We propose two distributed algorithms for LDA. The first algorithm is a straightforward mapping of LDA to a distributed processor setting. In this algorithm processors concurrently perform Gibbs sampling over local data followed by a global update of topic counts. The algorithm is simple to implement and can be viewed as an approximation to Gibbs-sampled LDA. The second version is a model that uses a hierarchical Bayesian extension of LDA to directly account for distributed data. This model has a theoretical guarantee of convergence but is more complex to implement than the first algorithm. Our distributed algorithm for HDP takes the straightforward mapping approach, and merges newly-created topics either by matching or by topic-id. Using five real-world text corpora we show that distributed learning works well in practice. For both LDA and HDP, we show that the converged test-data log probability for distributed learning is indistinguishable from that obtained with single-processor learning. Our extensive experimental results include learning topic models for two multi-million document collections using a 1024-processor parallel computer.

Keywords: topic models, latent Dirichlet allocation, hierarchical Dirichlet processes, distributed parallel computation

1. Introduction

Very large data sets, such as collections of images or text documents, are becoming increasingly common, with examples ranging from collections of online books at Google and Amazon, to the large collection of images at Flickr. These data sets present major opportunities for machine learning, such as the ability to explore richer and more expressive models than previously possible, and provide new and interesting domains for the application of learning algorithms.

However, the scale of these data sets also brings significant challenges for machine learning, particularly in terms of computation time and memory requirements. For example, a text corpus with one million documents, each containing one thousand words, will require approximately eight GBytes of memory to store the billion words. Adding the memory required for parameters, which usually exceeds memory for the data, creates a total memory requirement that exceeds that available

on a typical desktop computer. If one were to assume that a simple operation, such as computing a probability vector over categories using Bayes rule, takes on the order of 10^{-6} seconds per word, then a full pass through the billion words would take 1000 seconds. Thus, algorithms that make multiple passes through the data, for example clustering and classification algorithms, will have run times in days for this sized corpus. Furthermore, for small to moderate sized document sets where memory is not an issue, it would be useful to have algorithms that could take advantage of desktop multiprocessor/multicore technology to learn models in near real-time.

An obvious approach for addressing these time and memory issues is to distribute the learning algorithm over multiple processors. In particular, with P processors, it is somewhat trivial to address the memory issue by distributing $\frac{1}{P}$ of the total data to each processor. However, the computation problem remains non-trivial for a fairly large class of learning algorithms, namely how to combine local processing on each processor to arrive at a useful global solution.

In this general context we investigate distributed algorithms for two widely-used unsupervised learning models: the Latent Dirichlet Allocation (LDA) model, and the Hierarchical Dirichet Process (HDP) model. LDA and HDP models are arguably among the most successful recent learning algorithms for analyzing discrete data such as bags of words from a collection of text documents. However, they can take days to learn for large corpora, and thus, distributed learning would be particularly useful.

The rest of the paper is organized as follows: In Section 2 we review the standard derivation of LDA and HDP. Section 3 presents our two distributed algorithms for LDA and one distributed algorithm for HDP. Empirical results are provided in Section 4. Scalability results are presented in Section 5, and further analysis of distributed LDA is provided in Section 6. A comparison with related models is given in Section 7. Finally, Section 8 concludes the paper.

2. Latent Dirichlet Allocation and Hierarchical Dirichlet Process Model

We start by reviewing the LDA and HDP models. Both LDA and HDP are generative probabilistic models for discrete data such as bags of words from text documents—in this context these models are often referred to as topic models. To illustrate the notation, we refer the reader to the graphical models for LDA and HDP shown in Figure 1.

LDA models each of D documents in a collection as a mixture over K latent topics, with each topic being a multinomial distribution over a vocabulary of W words. For document j, we first draw a mixing proportion θ_j from a Dirichlet with parameter α. For the i^{th} word in the document, a topic $z_{ij} = k$ is drawn with probability $\theta_{k|j}$. Word x_{ij} is then drawn from topic z_{ij}, with x_{ij} taking on value w with probability $\phi_{w|z_{ij}}$. A Dirichlet prior with parameter β is placed on the word-topic distributions ϕ_k.

Thus, the generative process for LDA is given by

$$\theta_j \sim \mathcal{D}[\alpha], \quad \phi_k \sim \mathcal{D}[\beta], \quad z_{ij} \sim \theta_j, \quad x_{ij} \sim \phi_{z_{ij}}. \tag{1}$$

To avoid clutter we denote sampling from a Dirichlet $\theta_j \sim \mathcal{D}[\alpha]$ as shorthand for $[\theta_{1|j}, \ldots, \theta_{K|j}] \sim \mathcal{D}[\alpha, \ldots, \alpha]$, and likewise for ϕ. In this paper, we use symmetric Dirichlet priors for simplicity, unless specified otherwise. The full joint distribution over all parameters and variables is

$$p(\mathbf{x}, \mathbf{z}, \theta, \phi | \alpha, \beta) = \prod_j \frac{\Gamma(K\alpha)}{\Gamma(\alpha)^K} \prod_k \theta_{k|j}^{N_{kj}+\alpha-1} \prod_k \frac{\Gamma(W\beta)}{\Gamma(\beta)^W} \prod_w \phi_{w|k}^{N_{wk}+\beta-1}, \tag{2}$$

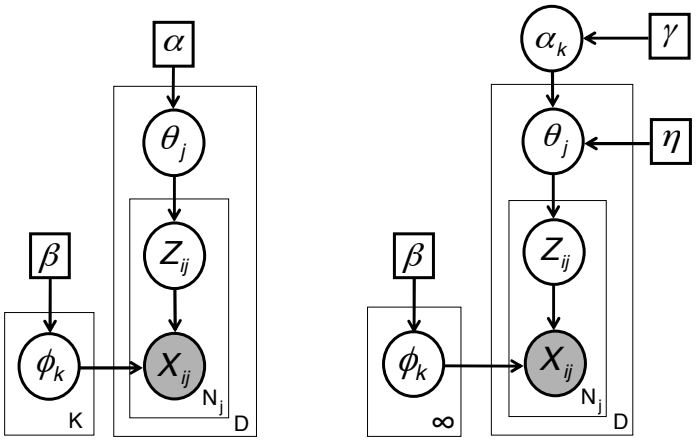


Figure 1: Graphical models for LDA (left) and HDP (right). Observed variables (words) are shaded, and hyperparameters are shown in squares.

	Description
D	Number of documents in collection
W	Number of distinct words in vocabulary
N	Total number of words in collection
K	Number of topics
x_{ij}	i^{th} observed word in document j
z_{ij}	Topic assigned to x_{ij}
N_{wk}	Count of word assigned to topic
N_{kj}	Count of topic assigned in document
ϕ_k	Probability of word given topic k
θ_j	Probability of topic given document j

Table 1: Description of commonly used variables.

where $N_{wkj} = \#\{i : x_{ij} = w, z_{ij} = k\}$, and we use the convention that missing indices are summed out. $N_{kj} = \sum_w N_{wkj}$ and $N_{wk} = \sum_j N_{wkj}$ are the two primary count arrays used in computations, representing the number of words assigned to topic k in document j, and the number of times word w is assigned to topic k in the corpus, respectively. For ease of reading we list commonly used variables in Table 1.

Given the observed words $\mathbf{x} = \{x_{ij}\}$, the task of Bayesian inference for LDA is to compute the posterior distribution over the latent topic assignments $\mathbf{z} = \{z_{ij}\}$, the mixing proportions θ_j, and the topics ϕ_k. Approximate inference for LDA can be performed either using variational methods (Blei et al., 2003) or Markov chain Monte Carlo methods (Griffiths and Steyvers, 2004). In this paper we focus on Markov chain Monte Carlo algorithms for approximate inference. MCMC is widely used as an inference method for a variety of topic models, for example Rosen-Zvi et al. (2004), Li and McCallum (2006), and Chemudugunta et al. (2007) all use MCMC for inference. In the MCMC context, the usual procedure is to integrate out the mixtures θ and topics ϕ in (2)—a procedure called collapsing—and just sample the latent variables $\mathbf{z}$. Given the current state of all but one variable z_{ij},

the conditional probability of z_{ij} is

$$p(z_{ij} = k|\mathbf{z}^{\neg ij}, \mathbf{x}, \alpha, \beta) \propto \frac{N_{wk}^{\neg ij} + \beta}{\sum_w N_{wk}^{\neg ij} + W\beta} \left(N_{kj}^{\neg ij} + \alpha \right), \tag{3}$$

where the superscript $\neg ij$ means that the corresponding word is excluded in the counts.

HDP is a collection of Dirichlet Processes which share the same topic distributions and can be viewed as the non-parametric extension of LDA. The advantage of HDP is that the number of topics is determined by the data. The HDP model is obtained by taking the following model in the limit as K goes to infinity. Let α_k be top level Dirichlet variables sampled from a Dirichlet with parameter γ/K. The topic mixture for each document, θ_j, is drawn from a Dirichlet with parameters $\eta\alpha_k$. The word-topic distributions ϕ_k are drawn from a base Dirichlet distribution with parameter β. As in LDA, z_{ij} is sampled from θ_j, and word x_{ij} is sampled from the corresponding topic $\phi_{z_{ij}}$. The generative process is given by

$$\alpha_k \sim \mathcal{D}[\gamma/K], \qquad \theta_j \sim D[\eta\alpha_k], \qquad \phi_k \sim D[\beta], \qquad z_{ij} \sim \theta_j, \qquad x_{ij} \sim \phi_{z_{ij}}.$$

The posterior distribution is sampled using the direct assignment sampler for HDP described in Teh et al. (2006). As was done for LDA, both θ and ϕ are integrated out, and z_{ij} is sampled from the following conditional distribution:

$$p(z_{ij} = k|\mathbf{z}^{\neg ij}, \mathbf{x}, \alpha, \beta, \eta) \propto \begin{cases} \frac{N_{wk}^{\neg ij} + \beta}{\sum_w N_{wk}^{\neg ij} + W\beta} \left(N_{kj}^{\neg ij} + \eta\alpha_k \right), & \text{if } k \text{ previously used} \\ \frac{\eta\alpha_{\text{new}}}{W}, & \text{if } k \text{ is new.} \end{cases} \tag{4}$$

The sampling scheme for α_k is also detailed in Teh et al. (2006). Note that a small amount of probability mass proportional to α_{new} is reserved for the instantiation of new topics. While HDP is defined to have infinitely many topics, the sampling algorithm only instantiates topics as needed.

2.1 Need for Distributed Algorithms

One could argue that it is trivial to distribute non-collapsed Gibbs sampling, because sampling of z_{ij} can happen independently given θ_j and ϕ_k, and therefore can be done concurrently. In the non-collapsed Gibbs sampler, one samples z_{ij} given θ_j and ϕ_k, and then samples θ_j and ϕ_k given z_{ij}. Furthermore, if individual documents are not spread across different processors, one can marginalize over just θ_j, since θ_j is processor-specific. In this partially collapsed scheme, the latent variables z_{ij} on each processor can be concurrently sampled, where the concurrency is over processors.

Unfortunately, the non-collapsed and partially collapsed Gibbs samplers exhibit slow convergence due to the strong dependencies between the parameters and latent variables. Generally, we expect faster mixing as more variables are collapsed (Liu et al., 1994; Casella and Robert, 1996). Figure 2 shows, using one of the data sets used throughout our paper, that the log probability of test data (measured as perplexity, which is defined in Section 4) of the non-collapsed and partially collapsed samplers converges more slowly than the fully collapsed sampler.

The slow convergence of partially collapsed and non-collapsed Gibbs samplers motivates the need to devise distributed algorithms for fully collapsed Gibbs samplers. In the following section we introduce distributed topic modeling algorithms that take advantage of the benefits of collapsing both θ and ϕ.

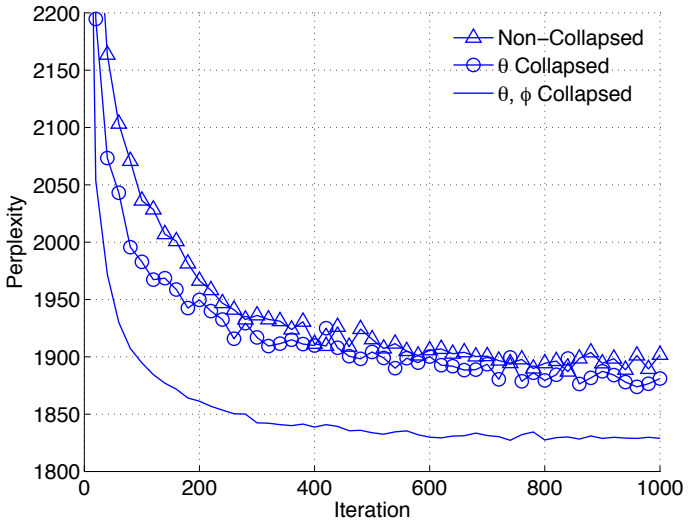


Figure 2: On the NIPS data set using $K = 20$ topics, the fully collapsed Gibbs sampler (solid line) converges faster than the partially collapsed (circles) and non-collapsed (triangles) samplers.

3. Distributed Algorithms for Topic Models

We introduce algorithms for LDA and HDP where the data, parameters, and computation are distributed over distinct processors. We distribute the D documents over P processors, with approximately $D_P = \frac{D}{P}$ documents on each processor. Documents are randomly assigned to processors, although as we will see later, the assignment of documents to processors—ranging from random to highly non-random or adversarial—appears to have little influence on the results. This indifference is somewhat understandable given that converged results from Gibbs sampling are independent of sampling order.

We partition the words from the D documents into $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_p, \ldots, \mathbf{x}_P\}$ and the corresponding topic assignments into $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_p, \ldots, \mathbf{z}_P\}$, where processor p stores $\mathbf{x}_p$, the words from documents $j = (p-1)D_P + 1, \ldots, pD_P$, and $\mathbf{z}_p$, the corresponding topic assignments. Topic-document counts N_{kj} are likewise distributed as N_{kjp}. The word-topic counts N_{wk} are also distributed, with each processor keeping a separate local copy N_{wkp}.

3.1 Approximate Distributed Latent Dirichlet Allocation

The difficulty of distributing and parallelizing over Gibbs sampling updates (3) lies in the fact that Gibbs sampling is a strictly sequential process. To asymptotically sample from the posterior distribution, the update of any topic assignment z_{ij} can not be performed concurrently with the update of any other topic assignment $z_{i'j'}$. But given the typically large number of word tokens compared to the number of processors, to what extent will the update of one topic assignment z_{ij} depend on the update of any other topic assignment $z_{i'j'}$? Our hypothesis is that this dependence is weak, and therefore we should be able to relax the requirement of sequential sampling of topic assignments and still learn a useful model. One can see this weak dependence in the following common situation. If two processors are concurrently sampling, but sampling different words in different documents

Algorithm 1 AD-LDA

repeat
  for each processor p in parallel **do**
    Copy global counts: $N_{wkp} \leftarrow N_{wk}$
    Sample $\mathbf{z}_p$ locally: LDA-Gibbs-Iteration($\mathbf{x}_p, \mathbf{z}_p, N_{kjp}, N_{wkp}, \alpha, \beta$)
  end for
  Synchronize
  Update global counts: $N_{wk} \leftarrow N_{wk} + \sum_p (N_{wkp} - N_{wk})$
until termination criterion satisfied

(i.e., $w_{ij} \neq w_{i'j'}$), then concurrent sampling will be very close to sequential sampling because the only term affecting the order of operations is the total count of topics $\sum_w N_{wk}$ in the denominator of (3).

The pseudocode for our Approximate Distributed LDA (AD-LDA) algorithm is shown in Algorithm 1. After distributing the data and parameters across processors, AD-LDA performs simultaneous LDA Gibbs sampling on each of the P processors. After processor p has swept through its local data and updated topic assignments $\mathbf{z}_p$, the processor has modified count arrays N_{kjp} and N_{wkp}. The topic-document counts N_{kjp} are distinct because of the document index, j, and will be consistent with the topic assignments $\mathbf{z}$. However, the word-topic counts N_{wkp} will in general be different on each processor, and not globally consistent with $\mathbf{z}$. To merge back to a single and consistent set of word-topic counts, we perform a reduce operation on N_{wkp} across all processors to update the global counts. After the synchronization and update operations, each processor has the same values in the N_{wkp} array which are consistent with the global vector of topic assignments $\mathbf{z}$. Note that N_{wkp} is not the result of P separate LDA models running on separate data. In particular, each word-topic count array reflects all the counts, not just those local to that processor, so for every processor $\sum_{wk} N_{wkp} = N$, where N is the total number of words in the corpus. As in LDA, the algorithm can terminate either after a fixed number of iterations, or based on some suitable MCMC convergence metric.

We chose the name *Approximate* Distributed LDA because in this algorithm we are no longer asymptotically sampling from the true posterior, but to an approximation of the true posterior. Nonetheless, we will show in our experimental results that the approximation made by Approximate Distributed LDA works very well.

3.2 Hierarchical Distributed Latent Dirichlet Allocation

In AD-LDA we constructed an algorithm where each processor is independently computing an LDA model, but at the end of each sweep through a processor's data, a consistent global array of topic counts N_{wk} is reconstructed. This global array of topic counts could be thought of as a parent topic distribution, from which each processor draws its own local topic distribution.

Using this intuition, we created a Bayesian model reflecting this structure, as shown in Figure 3. Our Hierarchical Distributed LDA model (HD-LDA) places a hierarchy over word-topic distributions, with Φ_k being the global or parent word-topic distribution and φ_{kp} the local word-topic distributions on each processor. The local word-topic distributions φ_{kp} are drawn from Φ_k according to a Dirichlet distribution with a topic-dependent strength parameter β_k, for each topic $k = 1 \ldots K$. The model on each processor is simply an LDA model. The generative process is given

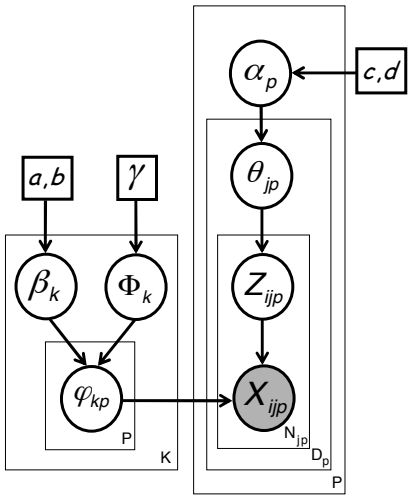


Figure 3: Graphical model for Hierarchical Distributed Latent Dirichlet Allocation.

by:

$$\beta_k \sim \mathcal{G}[a,b], \qquad \alpha_p \sim \mathcal{G}[c,d], \qquad \theta_{jp} \sim \mathcal{D}[\alpha_p], \qquad \Phi_k \sim \mathcal{D}[\gamma], \qquad \varphi_{kp} \sim \mathcal{D}[\beta_k \Phi_k], \qquad (5)$$
$$z_{ijp} \sim \theta_{jp}, \qquad x_{ijp} \sim \varphi_{z_{ijp}p}.$$

From this generative process, we derive Gibbs sampling equations for HD-LDA. The derivation is based on the Teh et al. (2006) sampling schemes for Hierarchical Dirichlet Processes. As was done for LDA, we start by integrating out φ and θ. The collapsed distribution of $\mathbf{z}_p$ and $\mathbf{x}_p$ on processor p is given by:

$$p(\mathbf{z}_p, \mathbf{x}_p | \alpha_p, \beta, \Phi) = \prod_j \left[\frac{\Gamma(K\alpha_p)}{\Gamma(N_{jp} + K\alpha_p)} \prod_k \frac{\Gamma(N_{kjp} + \alpha_p)}{\Gamma(\alpha_p)} \right] \prod_k \left[\frac{\Gamma(\beta_k)}{\Gamma(N_{kp} + \beta_k)} \prod_w \frac{\Gamma(N_{wkp} + \beta_k \Phi_{w|k})}{\Gamma(\beta_k \Phi_{w|k}))} \right]. \qquad (6)$$

From this we derive the conditional probability for sampling a topic assignment z_{ijp}. Unlike AD-LDA, the topic assignments on any processor are now conditionally independent of the topic assignments on the other processors given Φ, thus allowing each processor to sample $\mathbf{z}_p$ concurrently. The conditional probability of z_{ijp} is

$$p(z_{ijp} = k | \mathbf{z}_p^{\neg ijp}, \mathbf{x}, \alpha_p, \beta, \Phi) = (N_{kjp}^{\neg ijp} + \alpha_p) \frac{(N_{wkp}^{\neg ijp} + \beta_k \Phi_{w|k})}{(N_{kp}^{\neg ijp} + \beta_k)}.$$

The full derivation of the Gibbs sampling equations for HD-LDA is provided in Appendix A, which lists the complete set of sampling equations for α_p, β_k, and Φ_k.

The pseudocode for our Hierarchical Distributed LDA algorithm is given in Algorithm 2. Each variable in this model is either local or global, depending on whether inference for the variable is computed locally on a processor or globally, requiring information from all processors. Local variables include α, θ, φ, z, and x. Global variables include β and Φ. Each processor uses Gibbs

sampling to sample its local variables concurrently. After each sweep through the processor's data, the global variables are sampled. Note that, unlike AD-LDA, HD-LDA is performing strictly correct sampling for its model.

HD-LDA can be viewed as a mixture model with P LDA mixture components with equal mixing weights. In this view the data have been hard-assigned to their respective clusters (i.e., processors), and the parameters of the clusters are generated from a shared prior distribution.

Algorithm 2 HD-LDA

repeat
  for each processor p in parallel **do**
    Sample $\mathbf{z}_p$ locally: LDA-Gibbs-Iteration($\mathbf{x}_p, \mathbf{z}_p, N_{kjp}, N_{wkp}, \alpha_p, \beta_k \Phi_k$)
    Sample α_p locally
  end for
  Synchronize
  Sample: β_k, Φ_k
  Broadcast: β_k, Φ_k
until termination criterion satisfied

3.3 Approximate Distributed Hierarchical Dirichlet Processes

Our third distributed algorithm, Approximate Distributed HDP, takes the same approach as AD-LDA. Processors concurrently run HDP for a single sweep through their local data. After all of the processors sweep through their data, a synchronization and update step is performed to create a single set of globally-consistent word-topic counts N_{wk}. We refer to the distributed version of HDP as AD-HDP, and provide the pseudocode in Algorithm 3.

Unlike AD-LDA, which uses a fixed number of topics, individual processors in AD-HDP may instantiate new topics during the sampling phase, according to the HDP sampling Equation (4). During the synchronization and update step, instead of treating each processor's new topics as distinct, we merge new topics that were instantiated on different processors. Merging new topics helps limit unnecessary growth in the total number of topics and allows AD-HDP to produce more of a global model.

Algorithm 3 AD-HDP

repeat
  for each processor p in parallel **do**
    Sample z_p locally: HDP-Gibbs-Iteration($x_p, z_p, N_{kjp}, N_{wkp}, \alpha_{kp}, \beta, \gamma, \eta$)
    Report N_{wkp}, α_{kp} to master node
  end for
  Synchronize
  Update global counts (and merge new topics): $N_{wk} \leftarrow N_{wk} + \sum_p (N_{wkp} - N_{wk})$
  $\alpha_k \leftarrow (\sum_p \alpha_{kp})/P$
  Sample: η, α_k, γ
  Broadcast: $N_{wk}, \alpha_k, \gamma, \eta$
until termination criterion satisfied

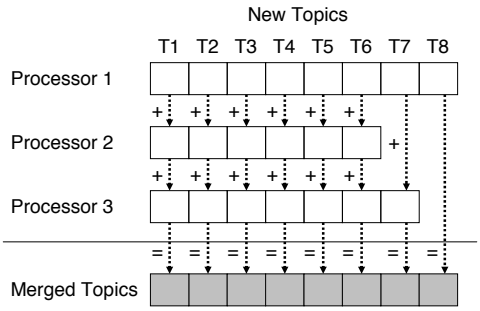


Figure 4: The simplest method to merge new topics in AD-HDP is by integer topic label.

There are several ways to merge newly created topics on each processor. A simple way—inspired by AD-LDA—is to merge new topics based on their integer topic label. A more complicated way is to match new topics across processors based on topic similarity.

In the first merging scheme, new topics are merged based on their integer topic label. For example, assume that we have three processors, and at the end of a sweep through the data, processor one has 8 new topics, processor two has 6 new topics, and processor three has 7 new topics. Then during synchronization, all these new topics would be aligned by topic label and their counts summed, producing 8 new global topics, as shown in Figure 4.

While this merging of new topics by topic-id may seem suboptimal, it is computationally simple and efficient. We will show in the next section that this merging generally works well in practice, even when processors only have a small amount of data. We suggest that even if the merging by topic-id is initially quite random, the subsequent dynamics align the topics in a sensible manner. We will also show that AD-HDP ultimately learns models with similar perplexity to HDP, irrespective of how new topics are merged.

We also investigate more complicated schemes for merging new topics in AD-HDP, beyond the simple approach of merging by topic-id. Instead of aligning new topics based topic-id it is possible to align new topics using a similarity metric such as symmetric Kullback-Leibler divergence. However, finding the optimal matching of topics in the case where $P > 2$ is NP-hard (Burkard and Çela, 1999). Thus, we consider approximate schemes: bipartite matching using a reference processor, and greedy matching.

In the bipartite matching scheme, we select a reference processor and perform bipartite matching between every processor's new topics and the set of new topics of the reference processor. The bipartite match is computed using the Hungarian algorithm, which runs in $O(T^3)$, producing an overall complexity of $O(PT^3)$ where T is the maximum number of new topics on a processor. We implemented this scheme but did not find any improvement over AD-HDP with merging by topic-id.

In the greedy matching scheme, new topics on each processor are sequentially compared to a global set of new topics. This global set is initialized to the first processor's set of new topics. If a new topic is sufficiently different from every topic in the global set, the number of topics in the global set is incremented; otherwise, the counts for that new topic are added to those from the closest match in the global set. A threshold is used to determine whether a new topic is sufficiently different from another topic. The worst case complexity of this algorithm is $O(P^2T^2)$—this is the case where every new topic is found to be different from every other new topic in the global set. Increasing this threshold will make it more likely for new topics to merge with the topics already in the global set (instead of incrementing the set), causing the expected running time of this merging algorithm to

Algorithm 4 Greedy Matching of New Topics for AD-HDP

```
Initialize global set of new topics, G, to be processor 1's set of new topics
for p = 2 to P do
  for topic t in processor p's set of new topics do
    Initialize score array
    for topic g in G do
      score[g] = symmetric-KL-divergence(t,g)
    end for
    if min(score) < threshold then
      Add t's counts to the topic in G corresponding to min(score)
    else
      Augment G with the new topic t
    end if
  end for
end for
```

	KOS	NIPS	WIKIPEDIA	PUBMED	NEWSGROUPS
D_{train}	3,000	1,500	2,051,929	8,200,000	19500
W	6,906	12,419	120,927	141,043	27,059
N	467,714	2,166,058	344,941,756	737,869,083	2,057,207
D_{test}	430	184	-	-	498

Table 2: Characteristics of data sets used in experiments.

be linear in the number of processors. The pseudocode of this greedy matching scheme is shown in Algorithm 4. This algorithm is run after each iteration of AD-HDP to produce a global set of new topics. We show in the next section that this greedy matching scheme significantly improves the rate of convergence for AD-HDP.

4. Experiments

The purpose of our experiments is to investigate how our distributed topic model algorithms, AD-LDA, HD-LDA and AD-HDP, perform when compared to their sequential counterparts, LDA and HDP. We are interested in two aspects of performance: the quality of the model learned, measured by log probability of test data; and the time taken to learn the model. Our primary data sets for these experiments were KOS blog entries, from dailykos.com, and NIPS papers, from books.nips.cc. We chose these relatively small data sets to allow us to perform a large number of experiments. Both data sets were split into a training set and a test set. Size parameters for these data sets are shown in Table 2. For each corpus, D is the number of documents, W is the vocabulary size and N is the total number of words. Two larger data sets WIKIPEDIA, from en.wikipedia.org, and PUBMED, from pubmed.gov were used for speedup experiments, described in Section 5. For precision-recall experiments we used the NEWSGROUPS data set, taken from the UCI Machine Learning Repository. All the data sets used in this paper can be downloaded from the UCI Machine Learning Repository (Asuncion and Newman, 2007).

Using the KOS and NIPS data sets, we computed test set perplexities for a range of topics K, and for numbers of processors, P, ranging from 1 to 3000. The distributed algorithms were initialized by first randomly assigning topics to words in $\mathbf{z}$, then counting topics in documents, N_{kjp}, and words in topics, N_{wkp}, for each processor. For each run of LDA, AD-LDA, and HD-LDA, a sample was taken at 500 iterations of the Gibbs sampler, which is well after the typical burn-in period of the initial 200-300 iterations. For each run of HDP and AD-HDP, we allow the Gibbs sampler to run for 3000 iterations, to allow the number of topics to grow. In our perplexity experiments, multiple processors were simulated in software by separating data, running sequentially through each processor, and simulating the global synchronization and update steps. For the speedup experiments, computations were run on 64 to 1024 processors on a 2000+ processor parallel supercomputer.

The following set of hyperparameters was used for the experiments, where hyperparameters are shown as variables in squares in the graphical models in Figures 1 and 3. For AD-LDA we set $\alpha = 0.1$ and $\beta = 0.01$. For AD-HDP we set $\beta = 0.01$, $\eta \sim \text{Gamma}(2,1)$ and $\gamma \sim \text{Gamma}(10,1)$. While η and γ could have also been fixed, resampling these hyperparameters allows for more robust topic growth, as described by Teh et al. (2006). For LDA and AD-LDA we fixed the hyperparameters α and β, but these priors could also be learned using sampling.

Selection of hyperparameters for HD-LDA was guided by our experience with AD-LDA. For AD-LDA, $\sum_w N_{wkp} \approx \frac{N}{K}$, but for HD-LDA $\sum_w N_{wkp} \approx \frac{N}{PK}$, so we choose a and b to make the mode of $\beta_k = \frac{(P-1)N}{PK}$ to simulate the inclusion of global counts in N_{wkp} as is done in AD-LDA. We set $\gamma = 2/K$, because it is important to scale γ by the number of topics to prevent oversmoothing when the counts are spread thinly among many topics. Finally, we choose c and d to make the mode of $\alpha_p = 0.1$, matching the value of α used in our LDA and AD-LDA experiments. Specifically, we set: $a = \frac{(P-1)N}{PK}$, $b = 1$, $c = 0.1 * 10 + 1$ and $d = 0.1$.

To systematically evaluate our distributed topic model algorithms, AD-LDA, HD-LDA and AD-HDP, we measured performance using test set perplexity, which is computed as $\text{Perp}(\mathbf{x}^{\text{test}}) = \exp(-\frac{1}{N^{\text{test}}} \log p(\mathbf{x}^{\text{test}}))$. For every test document, half the words at random are designated for fold-in, and the remaining words are used as test. The document mixture θ_j is learned using the fold-in part, and log probability of the test words is computed using this mixture, ensuring that the test words are not used in estimation of model parameters. For AD-LDA, the perplexity computation exactly follows that of LDA, since a single set of topic counts N_{wk} are saved when a sample is taken. In contrast, all P copies of N_{wkp} are required to compute perplexity for HD-LDA. Except where stated, perplexities are computed for all algorithms using $S = 10$ samples from the posterior from ten independent chains using

$$\log p(\mathbf{x}^{\text{test}}) = \sum_{j,w} N_{jw}^{\text{test}} \log \frac{1}{S} \sum_s \sum_k \theta_{k|j}^s \phi_{w|k}^s, \qquad \theta_{k|j}^s = \frac{\alpha + N_{kj}^s}{K\alpha + N_j^s}, \qquad \phi_{w|k}^s = \frac{\beta + N_{wk}^s}{W\beta + N_k^s}. \tag{7}$$

This perplexity computation follows the standard practice of averaging over multiple chains when making predictions with LDA models trained via Gibbs sampling, as discussed in Griffiths and Steyvers (2004). Averaging over ten samples significantly reduces perplexity compared to using a single sample from one chain. While we perform averaging over multiple samples to improve the estimate of perplexity, we have also observed similar relative results across our algorithms when we use a single sample to compute perplexity.

Analogous perplexity calculations are used for HD-LDA and AD-HDP. With HD-LDA we additionally compute processor-specific responsibilities, since test documents do not belong to any

particular processor, unlike the training documents. Each processor learns a document mixture θ_{jp} using the fold-in part for each test document. For each processor, the likelihood is calculated over the words in the fold-in part in a manner analogous to (7), and these likelihoods are normalized to form the responsibilities, r_p. To compute perplexity, we compute the likelihood over the test words, using a responsibility-weighted average of probabilities over all processors:

$$\log p(\mathbf{x}^{\text{test}}) = \sum_{j,w} N_{jw}^{\text{test}} \log \sum_p \frac{r_p}{S} \sum_s \sum_k \theta_{k|jp}^s \phi_{w|kp}^s$$

$$\text{where} \quad \theta_{k|jp}^s = \frac{\alpha_p + N_{kjp}^s}{K\alpha_p + N_{jp}^s}, \qquad \phi_{w|kp}^s = \frac{\beta_k \Phi_{w|k} + N_{wkp}^s}{\beta_k + N_{kp}^s}.$$

Computing perplexity in this manner prevents the possibility of seeing or using test words during the training and fold-in phases.

4.1 Perplexity

The perplexity results for KOS and NIPS in Figure 5 clearly show that the model perplexity is essentially the same for the distributed models AD-LDA and AD-HDP at $P = 10$ and $P = 100$ as their single-processor versions at $P = 1$. The figures show the test set perplexity, versus number of processors, P, for different numbers of topics K for the LDA-type models, and also for the HDP-models which learn the number of topics. The $P = 1$ perplexity is computed by LDA (circles) and HDP (triangles), and we use our distributed algorithms—AD-LDA (crosses), HD-LDA (squares), and AD-HDP (stars)—to compute the $P = 10$ and $P = 100$ perplexities. The variability in perplexity as a function of the number of topics is much greater than the variability due to the number of processors. Note that there is essentially no perplexity difference between AD-LDA and HD-LDA.

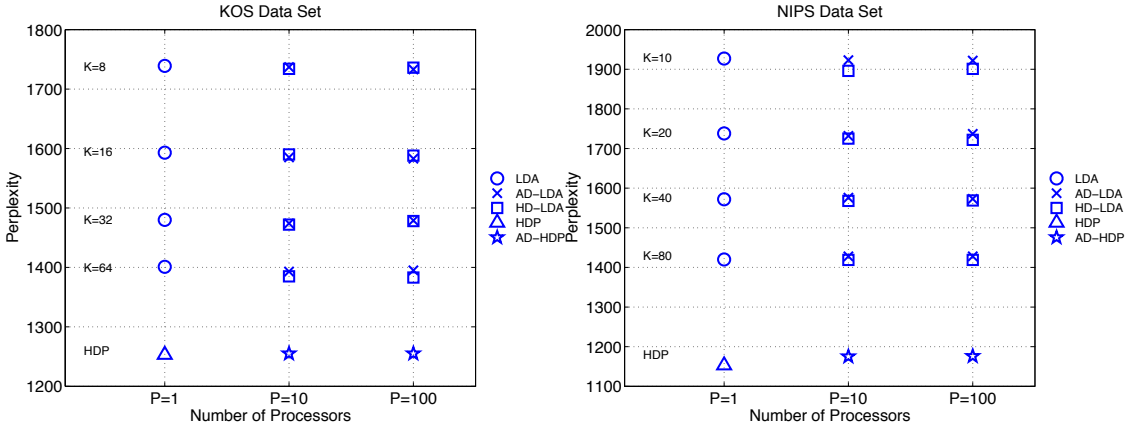


Figure 5: Test perplexity on KOS (left) and NIPS (right) data versus number of processors P. $P = 1$ corresponds to LDA and HDP. At $P = 10$ and $P = 100$ we show AD-LDA, HD-LDA and AD-HDP.

Even in the limit of a large number of processors, the perplexity for the distributed algorithms matches that for the sequential version. In fact, in the limiting case of just one document per processor, $P = 3000$ for KOS and $P = 1500$ for NIPS, we see that the perplexities of AD-LDA are generally no different to those of LDA, as shown in the rightmost point in each curve in Figure 6.

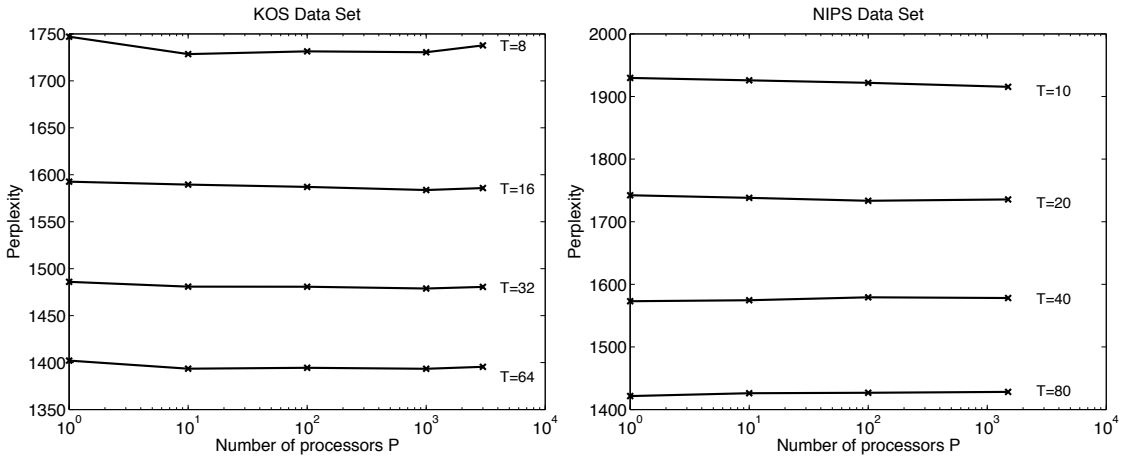


Figure 6: AD-LDA test perplexity versus number of processors up to the limiting case of number of processors equal to number of documents in collection. Left plot shows perplexity for KOS and right plot shows perplexity for NIPS.

AD-HDP instantiates fewer topics but produces a similar perplexity to HDP. The average number of topics instantiated by HDP on KOS was 669 while the average number of topics instantiated by AD-HDP was 490 ($P = 10$) and 471 ($P = 100$). For NIPS, HDP instantiated 687 topics while AD-HDP instantiated 569 ($P = 10$) and 569 ($P = 100$) topics. AD-HDP instantiates fewer topics because of the merging across processors of newly-created topics. The similar perplexity results for AD-HDP compared to HDP, despite the fewer topics, is partly due to the relatively small probability mass in many of the topics.

Despite no formal convergence guarantees, the approximate distributed algorithms, AD-LDA and AD-HDP, converged to good solutions in every single experiment (of the more than one hundred) we conducted using multiple real-world data sets. We also tested both our distributed LDA algorithms with adversarial/non-random distributions of topics across processors using synthesized data. One example of an adversarial distribution of documents is where each document only uses a single topic, and these documents are distributed such that processor p only has documents that are about topic p. In this case the distributed topic models have to learn the correct set of P topics, even though each processor only sees local documents that pertain to just one of the topics. We ran multiple experiments, starting with 1000 documents that were hard-assigned to $K = 10$ topics (i.e., each document is only about one topic), and distributing the 1000 documents over $P = 10$ processors, where each processor contained documents belonging to the same topic (an analogy is one processor only having documents about sports, the next processor only having documents about arts, and so on). The perplexity performance of AD-LDA and HD-LDA under these adversarial/non-random distribution of documents was as good as the performance when the documents were distributed randomly, and as good as the performance of single-processor LDA.

To demonstrate that the low perplexities obtained from the distributed algorithms with $P = 100$ processors are not just due to averaging effects, we split the NIPS corpus into one hundred 15-document collections, and ran LDA separately on each of these hundred collections. The test perplexity at $K = 40$ computed by averaging 100-separate LDA models was 2117, significantly

higher than the $P = 100$ test perplexity of 1575 for AD-LDA and HD-LDA. This shows that a baseline approach of simple averaging of results from separate processors performs much worse than the distributed coordinated learning algorithms that we propose in this paper.

4.2 Convergence

One could imagine that distributed algorithms, where each processor only sees its own local data, may converge more slowly than single-processor algorithms where the data is global. Consequently, we performed experiments to see whether our distributed algorithms were converging at the same rate as their sequential counterparts. If the distributed algorithms were converging slower, the computational gains of parallelization would be reduced. Our experiments consistently showed that the convergence rate for the distributed LDA algorithms was just as fast as those for the single processor case. As an example, Figure 7 shows test perplexity versus iteration of the Gibbs sampler for the NIPS data at $K = 20$ topics. During burn-in, up to iteration 200, the distributed algorithms are actually converging slightly faster than single processor LDA. Note that one iteration of AD-LDA or HD-LDA on a parallel multi-processor computer only takes a fraction (at best $\frac{1}{P}$) of the wall-clock time of one iteration of LDA on a single processor computer.

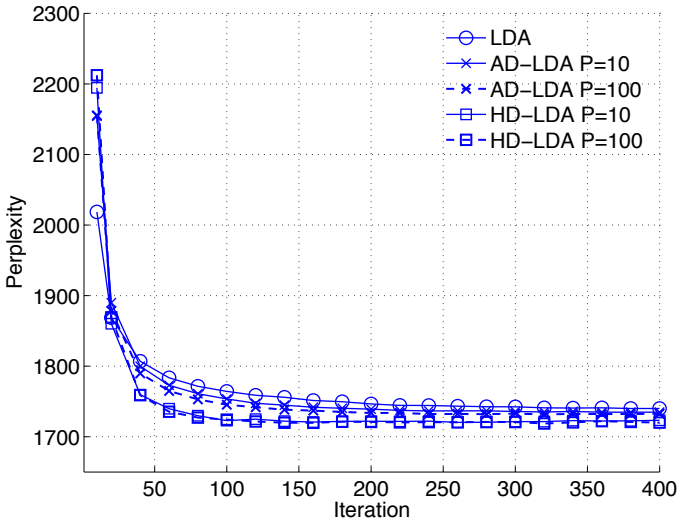


Figure 7: Convergence of test perplexity versus iteration for the distributed algorithms AD-LDA and HD-LDA using the NIPS data set and $K = 20$ topics.

We see slightly different convergence behavior in the non-parametric topic models. AD-HDP converges more slowly than HDP, as shown in Figure 8, due to AD-HDP's heavy averaging of new topics resulting from merging by topic-id (i.e., no matching). This slower convergence may partially be a result of the lower number of topics instantiated. The number of new topics instantiated in one pass of AD-HDP is limited to the maximum number of new topics instantiated on any one processor. For example, in the right plot, after 500 iterations, HDP has instantiated 360 topics, whereas AD-HDP has instantiated 210 ($P = 100$) and 250 ($P = 10$) topics. Correspondingly, at 500 iterations, the perplexity of HDP is lower than the perplexity of AD-HDP. After three thousand iterations, AD-HDP produces the same perplexity as HDP, which is reassuring because it indicates that AD-HDP

is ultimately producing a model that has the same predictive ability as HDP. We observe a similar result for the NIPS data set.

One way to accelerate the rate of convergence for AD-HDP is to match newly generated topics by similarity instead of by topic-id. Figure 9 shows that performing the greedy matching scheme for new topics as described in Algorithm 4 significantly improves the rate of convergence for AD-HDP. In this experiment, we used a threshold of 2 for determining topic similarity. The number of topics increases at a faster rate for AD-HDP with matching, since the greedy matching scheme is more flexible in that the number of new topics at each iteration is not limited to the maximum number of new topics instantiated on any one processor. The results show that the greedy matching scheme enables AD-HDP $P = 100$ to converge almost as quickly as HDP. In practice, only a few new topics are generated locally on each processor each iteration, and so the computational overhead of this heuristic matching scheme is minimal relative to the time for Gibbs sampling.

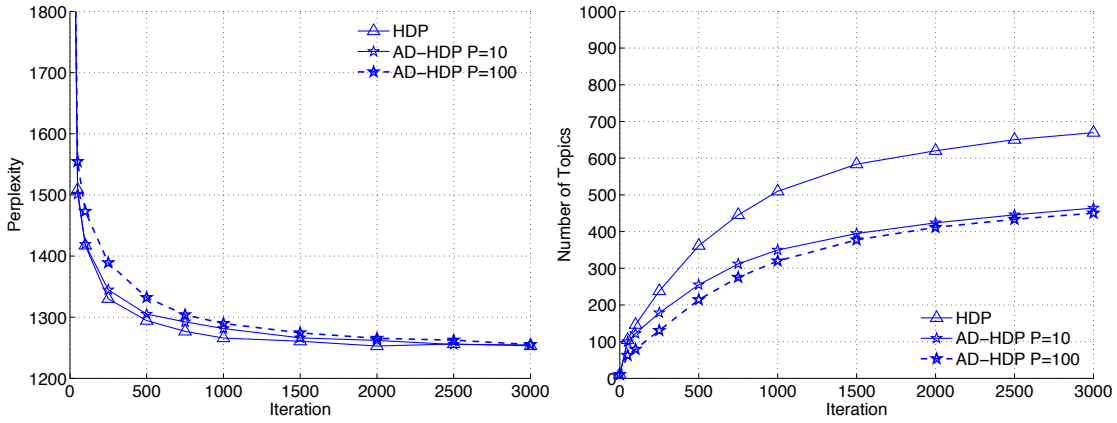


Figure 8: Results for HDP versus AD-HDP with no matching. Left plot shows test perplexity versus iteration for HDP and AD-HDP. Right plot shows number of topics versus iteration for HDP and AD-HDP. Results are for the KOS data set.

To further check that the distributed algorithms were performing comparably to their single processor counterparts, we ran experiments to investigate whether the results were sensitive to the number of topics used in the models, in case the distributed algorithms' performance worsens when the number of topics becomes very large. Figure 10 shows the test perplexity computed on the NIPS data set, as a function of the number of topics, for the LDA algorithms and a fixed number of processors $P = 10$ (the results for the KOS data set were quite similar and therefore not shown). The perplexities of the different algorithms closely track each other as number of topics, K, increases. In fact, in some cases HD-LDA produces slightly lower perplexities than those of single processor LDA. This lower perplexity may be due to the fact that in HD-LDA test perplexity is computed using P sets of topic parameters, thus it has more parameters than AD-LDA to better fit the data.

4.3 Precision and Recall

In addition to our experiments measuring perplexity, we also performed precision/recall calculations using the NEWSGROUPS data set, where each document's corresponding newsgroup is the class label. In this experiment we use LDA and AD-LDA to learn topic models on the training data. Once

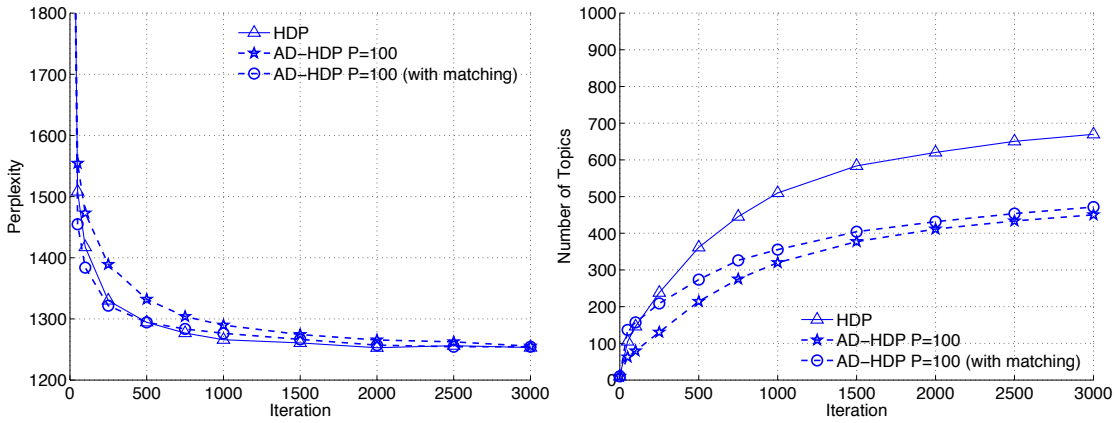


Figure 9: Results for HDP versus AD-HDP with greedy matching. Left plot shows test perplexity versus iteration for HDP and AD-HDP. Right plot shows number of topics versus iteration for HDP and AD-HDP. Results are for the KOS data set.

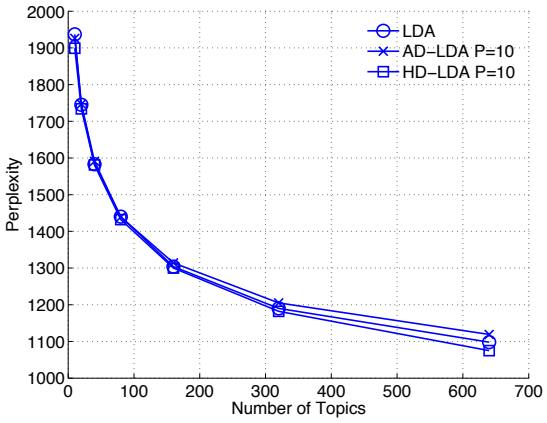


Figure 10: Test perplexity versus number of topics using the NIPS data set (S=5).

the model is learned, each test document can be treated as a "query", where the goal is to retrieve relevant documents from the training set. For each test document, the training documents are ranked according to how probable the test document is under each training document's mixture θ_j and the set of topics ϕ. From this ranking, one can calculate mean average precision and area under the ROC curve.

Figure 11 shows the mean average precision and the area under the ROC curve achieved by LDA and AD-LDA, plotted versus iteration. LDA performs slightly better than AD-LDA for the first 20 iterations, but AD-LDA catches up and converges to the same mean average precision and area under the ROC curve as LDA. This again shows that our distributed/parallel version of LDA produces a very similar result to the single-processor version.

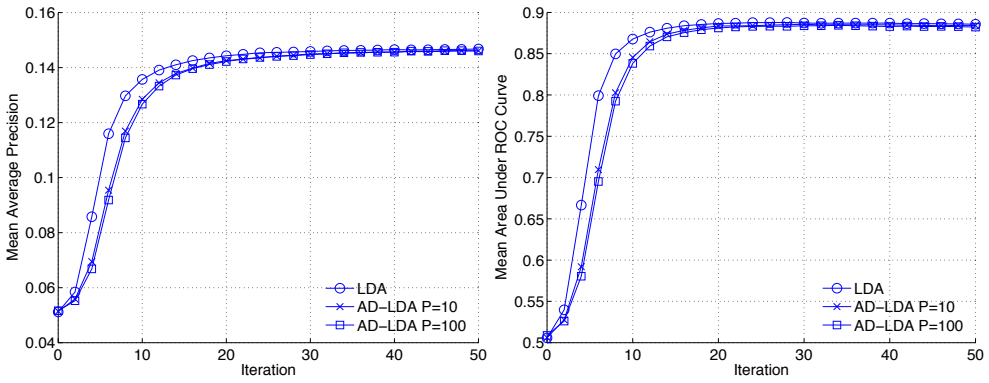


Figure 11: Precision/recall results: (left) Mean average precision for LDA/AD-LDA. (right) Area under the ROC curve for LDA/AD-LDA.

5. Scalability

The primary motivation for developing distributed algorithms for LDA and HDP is to have highly scalable algorithms, in terms of memory and computation time. Memory requirements depend on both memory for data and memory for model parameters. The memory for the data scales with N, the total number of words in the corpus. The memory for the parameters is linear in the number of topics K, which is either fixed for the LDA models or learned for the HDP models. The per-processor per-iteration time and space complexity of LDA and AD-LDA are shown in Table 3. AD-LDA's memory requirement scales well as collection sizes grow, because while corpus size (N and D) can get arbitrarily large, which can be offset by increasing the number of processors, P, the vocabulary size W will tend to asymptote, or at least grow more slowly. Similarly the time complexity scales well since the leading order term NK is divided by P.

The communication cost of the reduce operation, denoted by C in the table, represents the time taken to perform the global sum of the count difference $\sum_p (N_{wkp} - N_{wk})$. This is executed in $\log P$ stages and can be implemented efficiently in standard language/protocols such as MPI, the Message Passing Interface. Because of the additional KW term, parallel efficiency will depend on $\frac{N}{PW}$, with increasing efficiency as this ratio increases. Space and time complexity of HD-LDA are similar to that of AD-LDA, but HD-LDA has bigger constants. For a given number of topics, K, we argue that AD-HDP has similar time complexity as AD-LDA.

We performed large-scale speedup experiments with just AD-LDA instead of all three of our distributed topic modeling algorithms because AD-LDA produces very similar results to HD-LDA, but with significantly less computation. We expect that relative speedup performance for HD-LDA and AD-HDP should follow that for AD-LDA.

	LDA	AD-LDA
Space	$N+K(D+W)$	$\frac{1}{P}(N+KD)+KW$
Time	NK	$\frac{1}{P}NK+KW+C$

Table 3: Space and time complexity of LDA and AD-LDA.

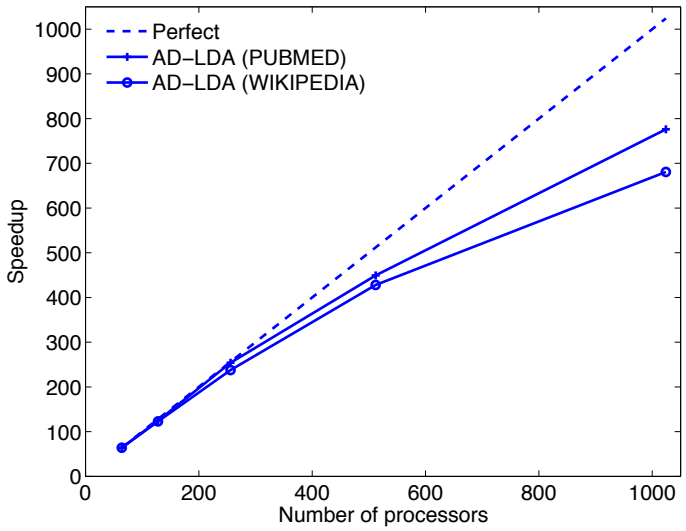


Figure 12: Parallel speedup results for 64 to 1024 processors on multi-million document data sets WIKIPEDIA and PUBMED.

We used two multi-million document data sets, WIKIPEDIA and PUBMED, for speedup experiments on a large-scale supercomputer. The supercomputer used was DataStar, a 15.6 TFlop terascale machine at San Diego Supercomputer Center built from 265 IBM P655 8-way compute nodes. We implemented a parallel version of AD-LDA using the Message Passing Interface protocol. We ran AD-LDA on WIKIPEDIA using $K = 1000$ topics and PUBMED using $K = 2000$ topics distributed over $P = 64, 128, 256, 512$ and 1024 processors. The speedup results, shown in Figure 12, show relatively high parallel efficiency, with approximately 700 times speedup for WIKIPEDIA and 800 times speedup for PUBMED when using $P = 1024$ processors, corresponding to parallel efficiencies of approximately 0.7 and 0.8 respectively. This speedup is computed relative to the time per iteration when using $P = 64$ processors (i.e., at $P = 64$ processors speedup=64), since it is not possible, due to memory limitations, to run these models on a single processor. Multiple runs were timed for both WIKIPEDIA and PUBMED, and the resulting variation in timing was less than 1%, so error bars are not shown in the figure. We see slightly higher parallel efficiency for PUBMED versus WIKIPEDIA because PUBMED has a larger amount of computation per unit data communicated, $\frac{N}{PW}$.

This speedup dramatically reduces the learning time for large topic models. If we were to learn a $K = 2000$ topic model for PUBMED using LDA on a single processor, it would require over 300 days instead of the 10 hours required to learn the same model using AD-LDA on 1024 processors. In our speedup experiments on these large data sets, we did not directly investigate latency or communication bandwidth effects. Nevertheless, one could expect that if the communication time becomes very long compared to the computation time, then it may be worth doing multiple Gibbs sampling sweeps on a processor's local data before performing the synchronization and global update step. In Section 6 we further examine this question of frequency of synchronizations. The relative time for communication versus computation also effects the weak scaling of parallelization, where the problem size increases linearly with the number of processors. We expect that parallel efficiency will be relatively constant for weak scaling since $\frac{N}{PW}$ is constant.

In addition to the large-scale speedup experiments run on the 1024-processor parallel supercomputer, we also performed small-scale speedup experiments for AD-HDP on an 8-node parallel cluster running MPI. Using the NIPS data set we measured parallel efficiencies of 0.75 and 0.5 for $P = 4$ and $P = 8$. The latter result on 8 processors means that the HDP model for NIPS can be learned four times faster than on a single processor.

6. Analysis of Approximate Distributed LDA

Finally, we investigate the dynamics of AD-LDA learning using toy data to get further insight into how AD-LDA is working. While we have shown experimental results showing that AD-LDA produces models with similar perplexity and similar convergence rates to LDA, it is not obvious why this algorithm works so well in practice. Our toy example has $W = 3$ words and $K = 2$ topics. We generated document collections according to the LDA generative process given by (1). We chose a low dimension vocabulary, W, so that we could plot the evolution of the Gibbs sampler on a two-dimensional word-topic simplex. We first generated data, then learned models using LDA and AD-LDA.

The left plot of Figure 13 shows the L_1 distance between the model's estimate of a particular topic-word distribution and the true distribution, as a function of Gibbs iteration, for both single-processor LDA and AD-LDA with $P = 2$. LDA and AD-LDA have qualitatively the same three-phase learning dynamics. The first four or so iterations (labeled *initialize*) correspond to somewhat random movement close to the randomly initialized starting point. In the next phase (labeled *burn-in*) both algorithms rapidly move in parameter space toward the posterior mode. And finally after burn-in (labeled *stationary*) both are sampling around the mode. In the right plot we show the similarity between AD-LDA and LDA samples taken from the equilibrium distribution—here plotted on the two-dimensional planar simplex corresponding to the three-word topic distribution.

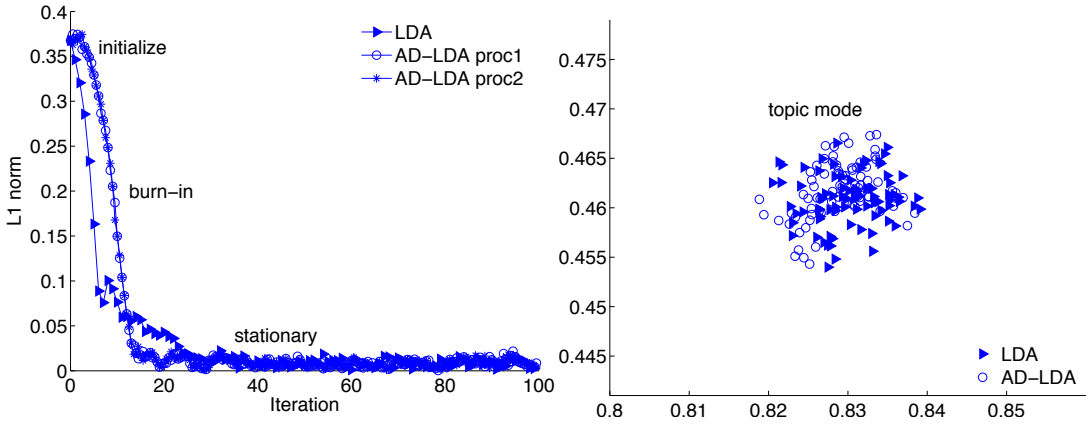


Figure 13: (Left) L_1 distance to the mode for LDA and for $P = 2$ AD-LDA. (Right) Closeup of 50 samples of ϕ (projected onto the topic simplex) taken from the equilibrium distribution, showing the similarity between LDA and $P = 2$ AD-LDA. Note the zoomed scale in this figure.

The left plot of Figure 14 depicts the same trajectory shown in Figure 13 left, projected onto the topic simplex. This plot shows the paths in parameter space of each model, and the same three-phase

learning dynamics: taking a few small steps near the starting point, moving up to the true solution, and then sampling near the posterior mode for the rest of the iterations. For each Gibbs iteration, the parameters corresponding to each of the two individual processors, and those parameters after merging, are shown for AD-LDA. One can see the alternating pattern of two separate (but close) parameter estimates on each processor, followed by a merged estimate. We observed that after the initial few iterations, the individual processor steps and the merge step each resulted in a move closer to the mode. One might worry that the AD-LDA algorithm would get trapped close to the initial starting point, for example, due to repeated label switching or oscillatory behavior of topic labeling across processors. In practice we have consistently observed that the algorithm quickly discards such configurations due to the stochastic nature of the moves and latches onto a consistent and stable labeling that rapidly moves it toward the posterior mode. The figure clearly illustrates that LDA and AD-LDA have qualitatively similar learning dynamics. The right plot in Figure 14 illustrates the same qualitative behavior as in the left plot, but now for $P = 10$ processors.

Interestingly, across a wide range of experiments, we observed that the variance in the AD-LDA word-topic distribution samples is typically only about 70% of the variance in LDA topic samples. Since the samplers are not the same it makes sense that the posterior variance differs (i.e., is underestimated) by the parallel sampler. We expect less variance because AD-LDA ignores fluctuations in the bulk of N_{wk}. Nonetheless, all of our experiments indicate that the posterior mode and means found by the parallel sampler are essentially the same as those found by the sequential sampler.

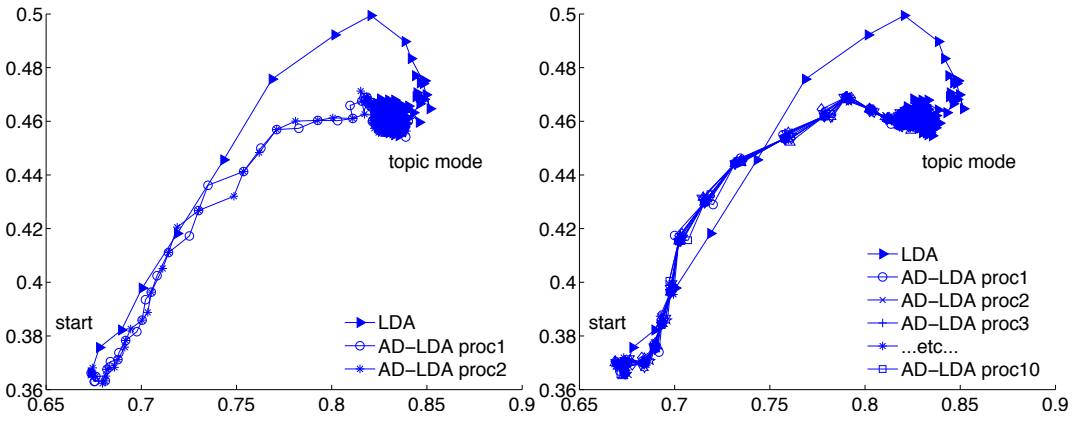


Figure 14: (Left) Projection of topics onto simplex, showing convergence to mode for $P = 2$. (Right) Same as left plot, but with $P = 10$.

Another insight can be gained by thinking of LDA as an approximation to stochastic descent in the space of assignment variables $\mathbf{z}$. On a single processor, one can view Gibbs sampling during burn-in as a stochastic algorithm to move up the likelihood surface. With multiple processors, each processor computes an upward direction in its own subspace, keeping all other directions fixed. The global update step then recombines these directions by vector-addition, in the same way as one would compute a gradient using finite differences. This is expected to be accurate as long as the surface is locally convex or concave, but will break down at saddle-points. We conjecture AD-LDA works reliably because saddle points are unstable and rare because the posterior is usually highly peaked for LDA models and high-dimensional count data sets.

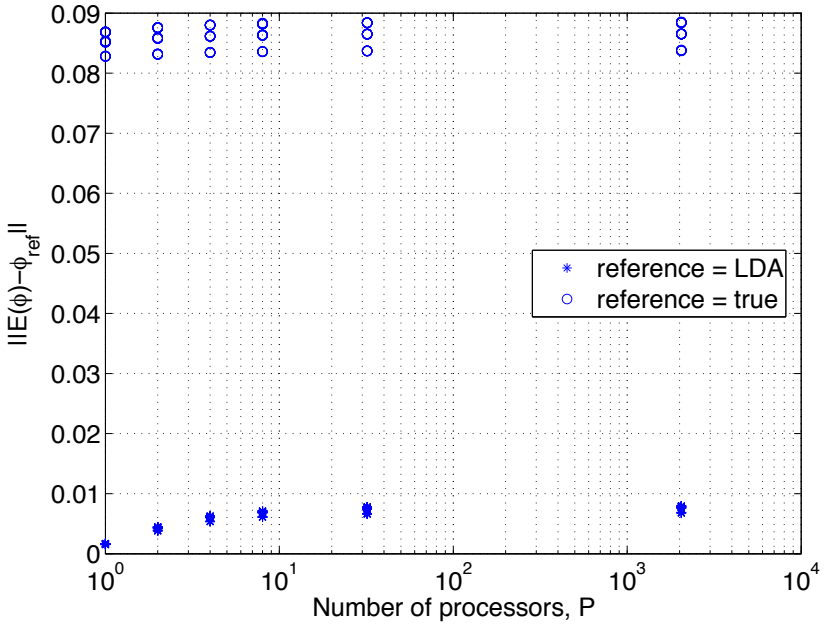


Figure 15: Average L_1 error in word-topic distribution versus P for AD-LDA.

While we see similar perplexities for AD-LDA compared to LDA, we could further ask if the AD-LDA algorithm is producing any bias in its estimates of the model parameters. To test this, we performed a series of experiments where we generated synthetic data sets according to the LDA generative process, with known word-topic distributions ϕ^*. We then learned LDA and AD-LDA models from each of the simulated data sets. We computed the expected value of the AD-LDA topics $E(\phi)$ and compared this to two reference values, ϕ_{ref} one based on the true distribution, $\phi_{\text{ref}} = \phi^*$, the other based on multiple LDA samples, $\phi_{\text{ref}} = E[\phi_{\text{LDA}}]$. Figure 15 shows that AD-LDA is much closer to the LDA topics $E[\phi_{\text{LDA}}]$ than either are to the true topics ϕ^*, telling us that the sampling variation in learning LDA models from finite data sets is much greater than the variation between LDA and AD-LDA on the same data sets.

6.1 When Does AD-LDA Fail?

In all of our experiments thus far, we have seen that our distributed algorithms learn models with equivalent predictive power as their non-distributed counterparts. However, when global synchronizations are done less frequently (i.e., when the synchronization step is performed after multiple Gibbs sampling sweeps through local data), the distributed algorithms may converge to suboptimal solutions.

When the synchronization interval is increased dramatically, it is possible for AD-LDA to converge to a suboptimal solution. This can happen because the topics (with the same integer label) on each processor can drift far apart, so that topic k on one processor diverges from topic k on another processor. In Figure 16, we show the results of an experiment on KOS where synchronizations only occur once every 100 iterations. For $P = 2$ processors, AD-LDA performs significantly worse than LDA. The $P = 2$ processor case is the worst case for AD-LDA, since one half of the total words on

each processor have the freedom to drift. In contrast, when $P = 100$ processors, each processor can only locally modify $1/100^{th}$ of the topic assignments, and so the topics on each processor can not drift far from the global set of topic counts at the previous iteration. Bipartite matching significantly improves the perplexity in the $P = 2$ processor case, suggesting that the lack of communication has indeed caused the topics to drift apart. Fortunately, topic drifting becomes less of a problem as more processors are used, and can be eliminated by frequent synchronization. It is also important to note that AD-LDA $P = 2$, where processors synchronize after every iteration, gives essentially identical results as LDA. Our recommendation in practice is to perform the synchronization and count updates after each iteration of the Gibbs sampler. As shown earlier in the paper, this leads to performance that is essentially indistinguishable from LDA. Since most multi-processor computing hardware will tend to have communication bandwidth matched to processor speed (i.e., faster and/or more processors usually come with a faster communication network), synchronizing after each iteration of the Gibbs sampler will usually be the optimal strategy.

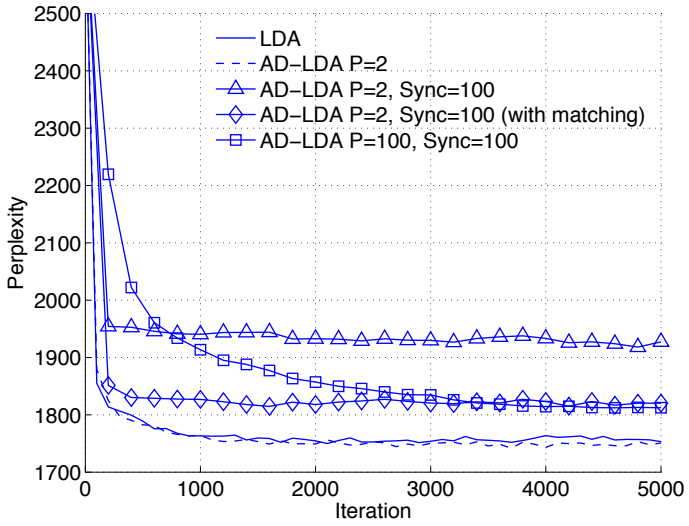


Figure 16: Test perplexity versus iteration where synchronizations between processors only occur every 100 iterations, KOS, $K = 16$.

7. Related Work

Approximate inference for topic models such as LDA and HDP can be carried out using a variety of methods, the most common being variational methods and Markov chain Monte Carlo methods. Previous efforts to parallelize these algorithms have focused on variational methods, which are often straightforward to cast in a distributed framework. For example, Blei et al. (2002) and Nallapati et al. (2007) describe distributed variational EM methods for LDA. In their distributed variational approach, the computationally expensive E-step is easily parallelized because the document-specific variational parameters are independent. Wolfe et al. (2008) investigate the parallelization of both the E and M-steps of variational EM for LDA, under a variety of computer network topologies. In these cases the distributed version of LDA produces identical results to the sequential version of the algorithm. However, memory for variational inference in LDA scales as MK, where M is

the number of distinct document-word pairs in the corpus. For typical English-language corpora, the total number of words in the corpus is less than twice the number of distinct document-word pairs ($N < 2M$), so M can be considered on the order of N. Since M is usually much larger than the number of documents, D, this memory requirement of MK is not nearly as scalable as that the memory requirement of $N + DK$ for MCMC methods.

Parallelized versions of various machine learning algorithms have also been developed. Forman and Zhang (2000) describe a parallel k-means algorithm, and W. Kowalczyk and N. Vlassis (2005) describe an asynchronous parallel EM algorithm for Gaussian mixture learning. A parallel EM algorithm for Probabilistic Latent Semantic Analysis, implemented using Google's MapReduce framework, was described in Das et al. (2007). A review of how to parallelize an array of standard machine learning algorithms using MapReduce was presented by Chu et al. (2007). Rossini et al. (2007) presents a framework for statisticians that allows for the parallel computing of independent tasks within the R language.

While many of these EM algorithms are readily parallelizable, Gibbs sampling of dependent variables (such as topic assignments) is fundamentally sequential and therefore difficult to parallelize. One way to parallelize Gibbs sampling is to run multiple independent chains in parallel to obtain multiple samples; however, this multiple-chain approach does not address the fact that the burn-in within each chain may take a long time. Furthermore, for some applications, one is not interested in multiple samples from independent chains. For example, if we wish to learn topics for a very large document collection, one is usually satisfied with mean values of word-topic distributions taken from a single chain.

One can parallelize a single MCMC chain by decomposing the variables into independent non-interacting blocks that can be sampled concurrently (Kontoghiorghes, 2005). However, when the variables are not independent, sampling variables in parallel is not possible. Brockwell (2006) presents a general parallel MCMC algorithm based on pre-fetching, but it is not practical for learning topic models because it discards most of its computations which makes it relatively inefficient. It is possible to construct partially parallel Gibbs samplers, in which the samples are independently accepted with some probability. In the limit as this probability goes to zero, this sampler will approach the sequential Gibbs sampler, as explained in P. Ferrari et al. (1993). However, this method is also not practical when learning topic models because it is computationally inefficient. Younes (1998) shows the existence of exact parallel samplers that make use of periodic synchronous random fields. However there is no known method for constructing such a sampler.

Our HD-LDA model is similar to the DCM-LDA model presented by Mimno and McCallum (2007). There the authors perform topic modeling on a collection of books by learning a different topic model for each book and then clustering these learned topics together to find global topics. In this model, the concept of a book is directly analogous to our concept of a processor. DCM-LDA uses Stochastic EM along with agglomerative clustering to learn topics, while our HD-LDA follows a fully Bayesian approach for inference. HD-LDA also differs from other topic hierarchies found in the literature. The Hierarchical Dirichlet Process model of Teh et al. (2006) places a deeper hierarchical prior on the topic mixture, instead of on the word-topic distributions. The Pachinko Allocation Model presented by Li and McCallum (2006) deals with a document-specific hierarchy of topic-assignments. These types of hierarchies do not directly facilitate proper parallel Gibbs sampling as is done in HD-LDA.

8. Conclusions

We have proposed three different algorithms for distributing across multiple processors Gibbs sampling for LDA and HDP. With our approximate distributed algorithm, AD-LDA, we sample from an approximation to the posterior distribution by allowing different processors to concurrently sample topic assignments on their local subsets of the data. Despite having no formal convergence guarantees, AD-LDA works very well empirically and is easy to implement. With our hierarchical distributed model, HD-LDA, we adapt the underlying LDA model to map to the distributed processor architecture. This model is more complicated than AD-LDA, but it inherits the usual convergence properties of Markov chain Monte Carlo. We discovered that careful selection of hyperparameters was critical to making HD-LDA work well, but this selection was clearly informed by AD-LDA. Our distributed algorithm AD-HDP followed the same approach as AD-LDA, but with an additional step to merge newly instantiated topics.

Our proposed distributed algorithms learn LDA models with predictive performance that is no different than single-processor LDA. On each processor they burn-in and converge at the same rate as LDA, yielding significant speedups in practice. For HDP, our distributed algorithm eventually produced the same perplexity as the single-processor version of HDP. Prior to reaching the converged perplexity result, AD-HDP had higher perplexity than HDP since the merging of new topics by label slows the rate of topic growth. We also discovered that matching new topics by similarity significantly improves AD-HDP's rate of convergence.

The space and time complexity of these distributed algorithms make them scalable to run very large data sets, for example, collections with billions to trillions of words. Using two multi-million document data sets, and running computations on a 1024-processor parallel supercomputer, we showed how one can achieve a 700-800 times reduction in wall-clock time by using our distributed approach.

There are several potentially interesting research directions that can be pursued using the algorithms proposed here as a starting point. One research direction is to use more complex schemes that allow data to adaptively move from one processor to another. The distributed schemes presented in this paper can also be used to parallelize topic models that are based on or derived from LDA and HDP, and beyond that a potentially larger class of graphical models.

Acknowledgments

This material is based upon work supported by the National Science Foundation. DN and PS were supported by NSF grants SCI-0225642, CNS-0551510, and IIS-0083489. DN was also supported by a Google Research Award. PS was also supported by ONR grant N00014-18-1-101 and a different Google Research Award. AA was supported by an NSF graduate fellowship. MW was supported by NSF grants IIS-0535278 and IIS-0447903, and ONR grant 00014-06-1-073. Computations were performed at SDSC under an MRAC Allocation.

Appendix A.

The auxiliary variable method explained in Escobar and West (1995) and Teh et al. (2006) is used to sample α, β, and Φ. To derive Gibbs sampling equations, we use the following expansions:

$$\frac{\Gamma(u)}{\Gamma(u+n)} = \frac{1}{\Gamma(n)} B(u,n) = \frac{1}{\Gamma(n)} \int_0^1 t^{u-1}(1-t)^{n-1}\,dt \tag{8}$$

$$\frac{\Gamma(u+n)}{\Gamma(u)} = \sum_{s=0}^{n} S(n,s)(u)^s \qquad \text{(S is Stirling number of first kind)} \tag{9}$$

The first expansion follows from the definition of the Beta function, and the second expansion makes use of the Stirling number of the first kind to rewrite the factorial (see Abramowitz and Stegun, 1964).

Now we derive the sampling equation for α_p. Combining the collapsed distribution (6) with the prior on α_p (5) gives the posterior distribution for α_p:[1]

$$P(\alpha_p|_-) \propto \prod_j \left[\frac{\Gamma(K\alpha_p)}{\Gamma(N_{jp}+K\alpha_p)} \prod_k \frac{\Gamma(N_{kjp}+\alpha_p)}{\Gamma(\alpha_p)}\right] \alpha_p^{c-1} e^{-d\alpha_p}.$$

Using the expansions (8,9) we introduce the auxiliary variables $\mathbf{t}$ and $\mathbf{s}$:

$$P(\alpha_p, \mathbf{t}, \mathbf{s}|_-) \propto \left[\prod_j t_j^{K\alpha_p - 1}(1-t_j)^{N_{jp}-1}\,dt_j\right] \left[\prod_j \prod_k S(N_{kjp}, s_{kjp}) \alpha_p^{s_{kjp}}\right] \alpha_p^{c-1} e^{-d\alpha_p}.$$

The joint distribution above allows us to create sampling equations for α_p, $\mathbf{t}$, and $\mathbf{s}$:

$$\begin{aligned} P(\alpha_p|\mathbf{t},\mathbf{s},_-) &\propto \left[\prod_j t_j^{K\alpha_p}\right] \left[\prod_j \prod_k \alpha_p^{s_{kjp}}\right] \alpha_p^{c-1} e^{-d\alpha_p} \\ &= \text{Gamma}\left[c + \sum_j \sum_k s_{kjp}; d - K\sum_j \log(t_j)\right], \end{aligned}$$

$$\begin{aligned} P(t_j|\alpha_p,\mathbf{s},_-) &\propto t_j^{K\alpha_p - 1}(1-t_j)^{N_{jp}-1} \\ &= \text{Beta}\,[K\alpha_p, N_{jp}], \end{aligned}$$

$$\begin{aligned} P(s_{kjp}|\alpha_p,\mathbf{t},_-) &\propto S(N_{kjp}, s_{kjp}) \alpha_p^{s_{kjp}} \\ &= \text{Antoniak}\,[N_{kjp}, \alpha_p]. \end{aligned}$$

The Antoniak distribution is the distribution of the number of occupied tables if N_{kjp} customers are sent into a restaurant that follows the Chinese restaurant process with strength parameter α_p. Sampling from the Antoniak distribution is done by sampling N_{kjp} Bernoulli variables:

1. To avoid notational clutter, we denote conditioned-upon variables and parameters by a dash. These variables can be inferred from context.

$$s^l_{kjp} \sim \text{Bernoulli}\left[\frac{\alpha_p}{\alpha_p + l - 1}\right] \qquad l = 1 \ldots N_{kjp},$$
$$s_{kjp} = \sum_l s^l_{kjp}.$$

Using the same auxiliary variable techniques, we derive sampling equations for β and Φ. These variables are sampled jointly because they are dependent. The posterior distribution for β and Φ and the joint distribution with the auxiliary variables $\mathbf{t}$ and $\mathbf{s}$ are given by:

$$P(\beta_k, \Phi_k|\_) \propto \prod_p \left[\frac{\Gamma(\beta_k)}{\Gamma(N_{kp}+\beta_k)} \prod_w \frac{\Gamma(N_{wkp}+\beta_k\Phi_{w|k})}{\Gamma(\beta_k\Phi_{w|k})}\right] \left[\prod_w \Phi^{\gamma-1}_{w|k}\right] \beta_k^{a-1} e^{-b\beta_k},$$
$$P(\beta_k, \Phi_k, \mathbf{t}, \mathbf{s}|\_) \propto \left[\prod_p t_{kp}^{\beta_k - 1}(1-t_{kp})^{N_{kp}-1}\right] \left[\prod_p \prod_w S(N_{wkp}, s_{wkp})(\beta_k\Phi_{w|k})^{s_{wkp}}\right]$$
$$\left[\prod_k \prod_w \Phi^{\gamma-1}_{w|k}\right] \beta_k^{a-1} e^{-b\beta_k}.$$

Note that the set of variables ($\mathbf{t}$ and $\mathbf{s}$) is unrelated to the set of auxiliary variables introduced for α_p. The sampling equations for β, Φ, $\mathbf{t}$, and $\mathbf{s}$ are:

$$P(\beta_k|\Phi, \mathbf{t}, \mathbf{s}, \_) \propto \left[\prod_p t_{kp}^{\beta_k}\right] \left[\prod_p \prod_w (\beta_k)^{s_{wkp}}\right] \beta_k^{a-1} e^{-b\beta_k}$$
$$= \text{Gamma}\left[a + \sum_p \sum_w s_{wkp}; b - \sum_p log(t_{kp})\right],$$

$$P(\Phi_k|\beta_k, \mathbf{t}, \mathbf{s}, \_) \propto \left[\prod_p \prod_w \Phi^{s_{wkp}}_{w|k}\right] \left[\prod_w \Phi^{\gamma-1}_{w|k}\right]$$
$$= \text{Dirichlet}\left[\gamma + \sum_p s_{wkp}\right],$$

$$P(t_{kp}|\beta_k, \Phi_k, \mathbf{s}, \_) \propto t_{kp}^{\beta_k - 1}(1-t_{kp})^{N_{kp}-1}$$
$$= \text{Beta}\left[\beta_k, N_{kp}\right],$$

$$P(s_{wkp}|\beta_k, \Phi_k, \mathbf{t}, \_) \propto S(N_{wkp}, s_{wkp})(\beta_k\Phi_{w|k})^{s_{wkp}}$$
$$= \text{Antoniak}\left[N_{wkp}, \beta_k\Phi_{w|k}\right].$$

References

M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, 1964.

A. Asuncion and D. Newman. UCI machine learning repository, 2007. URL http://www.ics.uci.edu/∼mlearn/MLRepository.html.

D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, volume 14, pages 601–608, Cambridge, MA, 2002. MIT Press.

D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

A. Brockwell. Parallel Markov chain Monte Carlo simulation by pre-fetching. *Journal of Computational & Graphical Statistics*, 15, No. 1:246–261, 2006.

R. Burkard and E. Çela. Linear assignment problems and extensions. In P. Pardalos and D. Du, editors, *Handbook of Combinatorial Optimization, Supplement Volume A*. Kluwer Academic Publishers, 1999.

G. Casella and C. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

C. Chemudugunta, P. Smyth, and M. Steyvers. Modeling general and specific aspects of documents with a probabilistic topic model. In *Advances in Neural Information Processing Systems 19*, pages 241–248. MIT Press, Cambridge, MA, 2007.

Chu, Kim, Lin, Yu, Bradski, Ng, and Olukotun. Map-reduce for machine learning on multicore. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 281–288. MIT Press, Cambridge, MA, 2007.

A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 271–280, New York, NY, 2007. ACM.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995. URL citeseer.ist.psu.edu/escobar94bayesian.html.

G. Forman and B. Zhang. Distributed data clustering can be efficient and exact. In *ACM KDD Explorations*, volume 2, pages 34–38, New York, NY, 2000. ACM.

T. Griffiths and M. Steyvers. Finding scientific topics. In *Proceedings of the National Academy of Sciences*, volume 101, pages 5228–5235, 2004.

E. Kontoghiorghes. *Handbook of Parallel Computing and Statistics (Statistics, Textbooks and Monographs)*. Chapman & Hall / CRC, 2005.

W. Li and A. McCallum. Pachinko allocation: DAG-structured mixture models of topic correlations. In *Proceedings of the International Conference on Machine Learning*, volume 23, pages 577–584, New York, NY, 2006. ACM.

J. Liu, W. Wong, and A. Kong. Covariance structure of the Gibbs sampler with applications to the comparisons of estimators and augmentation schemes. *Biometrika*, 81(1):27–40, 1994.

D. Mimno and A. McCallum. Organizing the OCA: Learning faceted subjects from a library of digital books. In *JCDL '07: Proceedings of the 2007 conference on digital libraries*, pages 376–385, New York, NY, 2007. ACM.

R. Nallapati, W. Cohen, and J. Lafferty. Parallelized variational EM for latent Dirichlet allocation: An experimental evaluation of speed and scalability. In *ICDMW '07: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops*, pages 349–354, Washington, DC, 2007. IEEE Computer Society.

P. Ferrari, A. Frigessi, and R. Schonmann. Convergence of some partially parallel Gibbs samplers with annealing. In *Annals of Applied Probability*, volume 3, pages 137–153. Institute of Mathematical Statistics, 1993.

M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, volume 20, pages 487–494, Arlington, VA, 2004. AUAI Press.

A. Rossini, L. Tierney, and N. Li. Simple parallel statistical computing in R. *Journal of Computational & Graphical Statistics*, 16(2):399, 2007.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

W. Kowalczyk and N. Vlassis. Newscast EM. In *Advances in Neural Information Processing Systems 17*, pages 713–720. MIT Press, Cambridge, MA, 2005.

J. Wolfe, A. Haghighi, and D. Klein. Fully distributed EM for very large datasets. In *Proceedings of the International Conference on Machine Learning*, pages 1184–1191. ACM, New York, NY, 2008.

L. Younes. Synchronous random fields and image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):380–390, 1998.

Journal of Machine Learning Research 10 (2009) 1829-1850 Submitted 5/08; Revised 11/08; Published 8/09

Nonlinear Models Using Dirichlet Process Mixtures

Babak Shahbaba BABAKS@UCI.EDU
Department of Statistics
University of California at Irvine
Irvine, CA, USA

Radford Neal RADFORD@STAT.TORONTO.EDU
Departments of Statistics and Computer Science
University of Toronto
Toronto, ON, Canada

Editor: Zoubin Ghahramani

Abstract

We introduce a new nonlinear model for classification, in which we model the joint distribution of response variable, y, and covariates, x, non-parametrically using Dirichlet process mixtures. We keep the relationship between y and x linear within each component of the mixture. The overall relationship becomes nonlinear if the mixture contains more than one component, with different regression coefficients. We use simulated data to compare the performance of this new approach to alternative methods such as multinomial logit (MNL) models, decision trees, and support vector machines. We also evaluate our approach on two classification problems: identifying the folding class of protein sequences and detecting Parkinson's disease. Our model can sometimes improve predictive accuracy. Moreover, by grouping observations into sub-populations (i.e., mixture components), our model can sometimes provide insight into hidden structure in the data.

Keywords: mixture models, Dirichlet process, classification

1. Introduction

In regression and classification models, estimation of parameters and interpretation of results are easier if we assume that distributions have simple forms (e.g., normal) and that the relationship between a response variable and covariates is linear. However, the performance of such a model depends on the appropriateness of these assumptions. Poor performance may result from assuming wrong distributions, or regarding relationships as linear when they are not. In this paper, we introduce a new model based on a Dirichlet process mixture of simple distributions, which is more flexible in capturing nonlinear relationships.

A Dirichlet process, $\mathcal{D}(G_0, \gamma)$, with baseline distribution G_0 and scale parameter γ, is a distribution over distributions. Ferguson (1973) introduced the Dirichlet process as a class of prior distributions for which the support is large, and the posterior distribution is manageable analytically. Using the Polya urn scheme, Blackwell and MacQueen (1973) showed that the distributions sampled from a Dirichlet process are discrete almost surely.

The idea of using a Dirichlet process as the prior for the mixing proportions of a simple distribution (e.g., Gaussian) was first introduced by Antoniak (1974). In this paper, we will describe the Dirichlet process mixture model as a limit of a finite mixture model (see Neal, 2000, for further description). Suppose exchangeable random values $y_1, ..., y_n$ are drawn independently from some

unknown distribution. We can model the distribution of y as a mixture of simple distributions, with probability or density function

$$P(y) \quad = \quad \sum_{c=1}^{C} p_c F(y, \phi_c).$$

Here, p_c are the mixing proportions, and $F(y,\phi)$ is the probability or density for y under a distribution, $F(\phi)$, in some simple class with parameters ϕ—for example, a normal in which $\phi = (\mu, \sigma)$. We first assume that the number of mixing components, C, is finite. In this case, a common prior for p_c is a symmetric Dirichlet distribution, with density function

$$P(p_1, ..., p_C) \quad = \quad \frac{\Gamma(\gamma)}{\Gamma(\gamma/C)^C} \prod_{c=1}^{C} p_c^{(\gamma/C)-1},$$

where $p_c \geq 0$ and $\sum p_c = 1$. The parameters ϕ_c are independent under the prior, with distribution G_0. We can use mixture identifiers, c_i, and represent this model as follows:

$$\begin{aligned} y_i | c_i, \phi &\sim F(\phi_{c_i}), \\ c_i | p_1, ..., p_C &\sim \mathit{Discrete}(p_1, ..., p_C), \\ p_1, ..., p_C &\sim \mathit{Dirichlet}(\gamma/C, ..., \gamma/C), \\ \phi_c &\sim G_0. \end{aligned} \tag{1}$$

By integrating over the Dirichlet prior, we can eliminate the mixing proportions, p_c, and obtain the following conditional distribution for c_i:

$$P(c_i = c | c_1, ..., c_{i-1}) \quad = \quad \frac{n_{ic} + \gamma/C}{i - 1 + \gamma}. \tag{2}$$

Here, n_{ic} represents the number of data points previously (i.e., before the i^{th}) assigned to component c. As we can see, the above probability becomes higher as n_{ic} increases.

When C goes to infinity, the conditional probabilities (2) reach the following limits:

$$\begin{aligned} P(c_i = c | c_1, ..., c_{i-1}) &\rightarrow \frac{n_{ic}}{i - 1 + \gamma}, \\ P(c_i \neq c_j \text{ for all } j < i | c_1, ..., c_{i-1}) &\rightarrow \frac{\gamma}{i - 1 + \gamma}. \end{aligned}$$

As a result, the conditional probability for θ_i, where $\theta_i = \phi_{c_i}$, becomes

$$\theta_i | \theta_1, ..., \theta_{i-1} \quad \sim \quad \frac{1}{i - 1 + \gamma} \sum_{j<i} \delta_{\theta_j} + \frac{\gamma}{i - 1 + \gamma} G_0, \tag{3}$$

where δ_θ is a point mass distribution at θ. Since the observations are assumed to be exchangeable, we can regard any observation, i, as the last observation and write the conditional probability of θ_i given the other θ_j for $j \neq i$ (written θ_{-i}) as follows:

$$\theta_i | \theta_{-i} \quad \sim \quad \frac{1}{n - 1 + \gamma} \sum_{j \neq i} \delta_{\theta_j} \; + \; \frac{\gamma}{n - 1 + \gamma} G_0. \tag{4}$$

The above conditional probabilities are equivalent to the conditional probabilities for θ_i according to the Dirichlet process mixture model (as presented by Blackwell and MacQueen, 1973, using the Polya urn scheme), which has the following form:

$$\begin{aligned} y_i|\theta_i &\sim F(\theta_i), \\ \theta_i|G &\sim G, \\ G &\sim \mathcal{D}(G_0,\gamma). \end{aligned} \tag{5}$$

That is, If we let $\theta_i = \phi_{c_i}$, the limit of model (1) as $C \to \infty$ becomes equivalent to the Dirichlet process mixture model (Ferguson, 1983; Neal, 2000). In model (5), G is the distribution over θ's, and has a Dirichlet process prior, $\mathcal{D}$. Phrased this way, each data point, i, has its own parameters, θ_i, drawn independently from a distribution that is drawn from a Dirichlet process prior. But since distributions drawn from a Dirichlet process are discrete (almost surely) as shown by Blackwell and MacQueen (1973), the θ_i for different data points may be the same.

The parameters of the Dirichlet process prior are G_0, a distribution from which θ's are sampled, and γ, a positive scale parameter that controls the number of components of the mixture that will be represented in the sample, such that a larger γ results in a larger number of components. To illustrate the effect of γ on the number of mixture components in a sample of size 200, we generated samples from four different Dirichlet process priors with $\gamma = 0.1, 1, 5, 10$, and the same baseline distribution $G_0 = N_2(0, 10I_2)$ (where I_2 is a 2×2 identity matrix). For a given value of γ, we first sample θ_i, where $i = 1, ..., 200$, according to the conditional probabilities (3), and then we sample $y_i|\theta_i \sim N_2(\theta_i, 0.2I_2)$. The data generated according to these priors are shown in Figure 1. As we can see, the (prior) expected number of components in a finite sample increases as γ becomes larger.

With a Dirichlet process prior, we can we find conditional distributions of the posterior distribution of model parameters by combining the conditional prior probability of (4) with the likelihood $F(y_i, \theta_i)$, obtaining

$$\theta_i|\theta_{-i}, y_i \sim \sum_{j \neq i} q_{ij}\delta_{\theta_j} + r_i H_i, \tag{6}$$

where H_i is the posterior distribution of θ based on the prior G_0 and the single data point y_i, and the values of the q_{ij} and r_i are defined as follows:

$$\begin{aligned} q_{ij} &= bF(y_i, \theta_j), \\ r_i &= b\gamma \int F(y_i, \theta) dG_0(\theta). \end{aligned}$$

Here, b is such that $r_i + \sum_{j \neq i} q_{ij} = 1$. These conditional posterior distributions are what are needed when sampling the posterior using MCMC methods, as discussed further in Section 2.

Bush and MacEachern (1996), Escobar and West (1995), MacEachern and Müller (1998), and Neal (2000) have used Dirichlet process mixture models for density estimation. Müller et al. (1996) used this method for curve fitting. They model the joint distribution of data pairs (x_i, y_i) as a Dirichlet process mixture of multivariate normals. The conditional distribution, $P(y|x)$, and the expected value, $E(y|x)$, are estimated based on this distribution for a grid of x's (with interpolation) to obtain a nonparametric curve. The application of this approach (as presented by Müller et al., 1996) is restricted to continuous variables. Moreover, this model is feasible only for problems with a small number of covariates, p. For data with moderate to large dimensionality, estimation of the joint

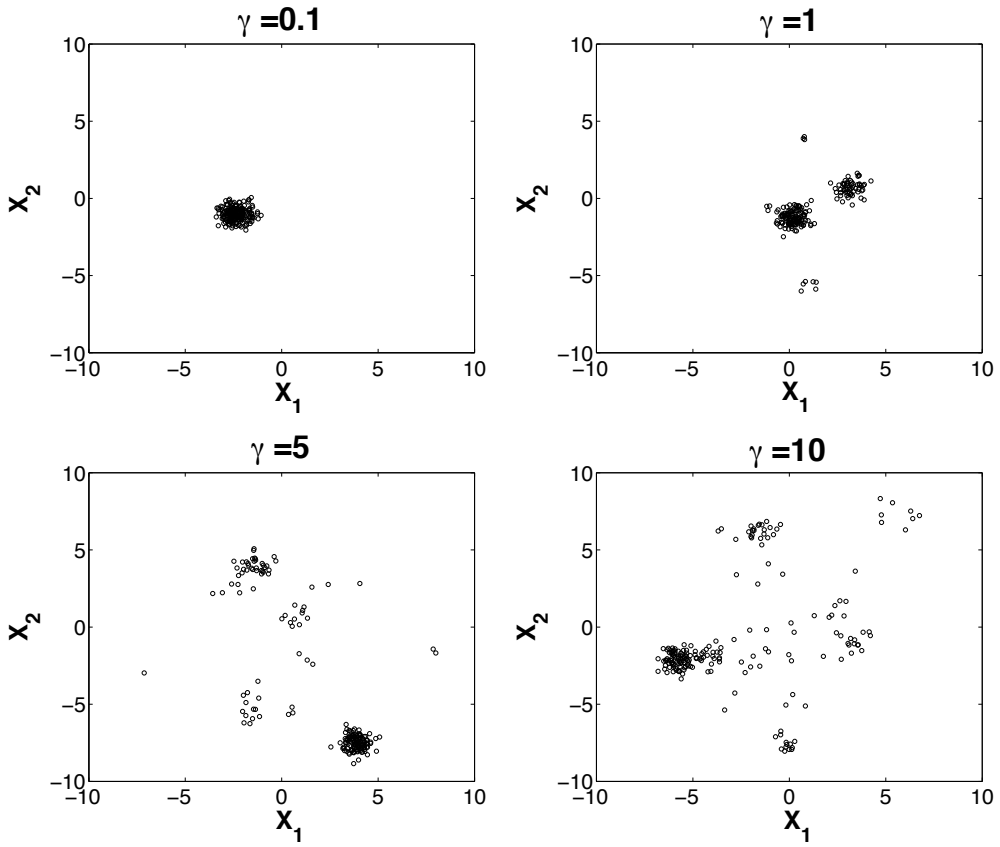


Figure 1: Data sets of size $n = 200$ generated according to four different Dirichlet process mixture priors, each with the same baseline distribution, $G_0 = N_2(0, 10I_2)$, but different scale parameters, γ. As γ increases, the expected number of components present in the sample becomes larger. (Note that, as can be seen above, when γ is large, many of these components have substantial overlap.)

distribution is very difficult both statistically and computationally. This is mostly due to the difficulties that arise when simulating from the posterior distribution of large full covariance matrices. In this approach, if a mixture model has C components, the set of full covariance matrices have $Cp(p+1)/2$ parameters. For large p, the computational burden of estimating these parameters might be overwhelming. Estimating full covariance matrices can also cause statistical difficulties since we need to assure that covariance matrices are positive semidefinite. Conjugate priors based the inverse Wishart distribution satisfy this requirement, but they lack flexibility (Daniels and Kass, 1999). Flat priors may not be suitable either, since they can lead to improper posterior distributions, and they can be unintentionally informative (Daniels and Kass, 1999). A common approach to address these issues is to use decomposition methods in specifying priors for full covariance matrices (see for example, Daniels and Kass, 1999; Cai and Dunson, 2006). Although this approach has demonstrated some computational advantages over direct estimation of full covariance matrices, it is not yet feasible for high-dimensional variables. For example, Cai and Dunson (2006) recommend their approach only for problems with less than 20 covariates.

We introduce a new nonlinear Bayesian model, which also nonparametrically estimates $P(x,y)$, the joint distribution of the response variable y and covariates x, using Dirichlet process mixtures. Within each component, we assume the covariates are independent, and model the dependence between y and x using a linear model. Therefore, unlike the method of Müller et al. (1996), our approach can be used for modeling data with a large number of covariates, since the covariance matrix for one mixture component is highly restricted. Using the Dirichlet process as the prior, our method has a built-in mechanism to avoid overfitting since the complexity of the nonlinear model is controlled. Moreover, this method can be used for categorical as well as continuous response variables by using a generalized linear model instead of the linear model.

The idea of building a nonlinear model based on an ensemble of simple linear models has been explored extensively in the field of machine learning. Jacobs et al. (1991) introduced a supervised learning procedure for models that are comprised of several local models (experts) each handling a subset of data. A gating network decides which expert should be used for a given data point. For inferring the parameters of such models, Waterhouse et al. (1996) provided a Bayesian framework to avoid over-fitting and noise level under-estimation problems associated with traditional maximum likelihood inference. Rasmussen and Ghahramani (2002) generalized mixture of experts models by using infinitely many nonlinear experts. In their approach, each expert is assumed to be a Gaussian process regression model, and the gating network is based on an input-dependent adaptation of Dirichlet process. Meeds and Osindero (2006) followed the same idea, but instead of assuming that the covariates are fixed, they proposed a joint mixture of experts model over covariates and response variable.

Our focus here is on classification models with a multi-category response, in which we have observed data for n cases, (x_1,y_1),...,(x_n,y_n). Here, the class y_i has J possible values, and the covariates x_i can in general be a vector of p covariates. We wish to classify future cases in which only the covariates are observed. For binary ($J = 2$) classification problems, a simple logistic model can be used, with class probabilities defined as follows (with the case subscript dropped from x and y):

$$P(y = 1|x,\alpha,\beta) \quad = \quad \frac{\exp(\alpha + x^T\beta)}{1+\exp(\alpha + x^T\beta)}.$$

When there are three or more classes, we can use a generalization known as the multinomial logit (MNL) model (called "softmax" in the machine learning literature):

$$P(y = j|x,\alpha,\beta) \quad = \quad \frac{\exp(\alpha_j + x^T\beta_j)}{\sum_{j'=1}^{J}\exp(\alpha_{j'} + x^T\beta_{j'})}.$$

MNL models are *discriminative*, since they model the conditional distribution $P(y|x)$, but not the distribution of covariates, $P(x)$. In contrast, our dpMNL model is *generative*, since it estimates the joint distribution of response and covariates, $P(x,y)$. The joint distribution can be decompose into the product of the marginal distribution $P(x)$ and the conditional distribution $P(y|x)$; that is, $P(x,y) = P(x)P(y|x)$.

Generative models have several advantages over discriminative models (see for example, Ulusoy and Bishop, 2005). They provide a natural framework for handling missing data or partially labeled data. They can also augment small quantities of expensive labeled data with large quantities of cheap unlabeled data. This is especially useful in applications such as document labeling and image analysis, where it may provide better predictions for new feature patterns not present in the data at

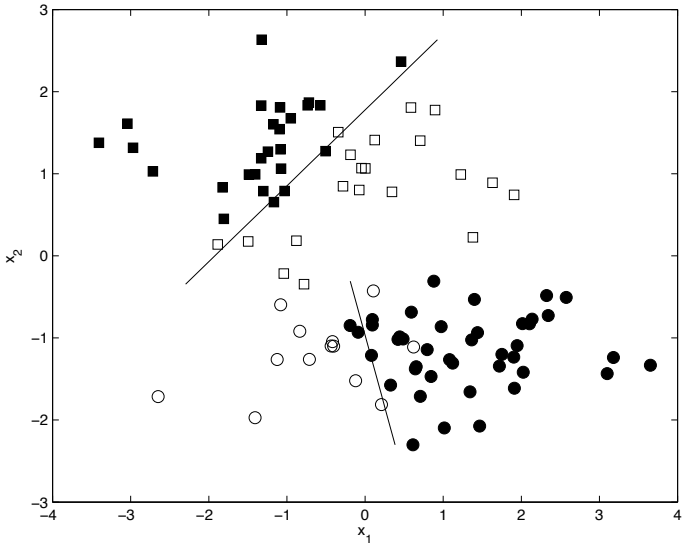


Figure 2: An illustration of our model for a binary (black and white) classification problem with two covariates. Here, the mixture has two components, which are shown with circles and squares. In each component, an MNL model separates the two classes into "black" or "white" with a linear decision boundary. The overall decision boundary, which is a smooth function, is not shown in this figure.

the time of training. For example, the Latent Dirichlet Allocation (LDA) model proposed by Blei et al. (2003) is a well-defined generative model that performs well in classifying documents with previously unknown patterns.

While generative models are quite successful in many problems, they can be computationally intensive. Moreover, finding a good (but not perfect) estimate for the joint distribution of all variables (i.e., x and y) does not in general guarantee a good estimate of decision boundaries. By contrast, discriminative models are often computationally fast and are preferred when the covariates are in fact non-random (e.g., they are fixed by an experimental design).

Using a generative model in our proposed method provides an additional benefit. Modeling the distribution of covariates jointly with y allows us to implicitly model the dependency of covariates on each other through clustering (i.e., assigning data points to different components), which could provide insight into hidden structure in the data. To illustrate this concept, consider Figure 2 where the objective is to classify cases into black or white. To improve predictive accuracy, our model has divided the data into two components, shown as squares and circles. These components are distinguished primarily by the value of the second covariate, x_2, which is usually positive for squares and negative for circles. For cases in the squares group, the response variable strongly depends on both x_1 and x_2 (the linear separator is almost diagonal), whereas, for cases in the circles group, the model mainly depends on x_1 alone (the linear model is almost vertical). Therefore, by grouping the data into sub-populations (e.g., circles and squares in this example), our model not only improves classification accuracy, but also discovers hidden structure in the data (i.e., by clustering covariate observations). This concept is briefly discussed in Section 5, where we use our model to predict Parkinson's disease. A more detailed discussion on using our method to detect hidden structure in

data is provided elsewhere (Shahbaba, 2009), where a Dirichlet process mixture of autoregressive models us used to analyze time-series processes that are subject to regime changes, with no specific economic theory about the structure of the model.

The next section describes our methodology. In Section 3, we illustrate our approach and evaluate its performance on simulated data. In Section 4, we present the results of applying our model to an actual classification problem, which attempts to identify the folding class of a protein sequence based on the composition of its amino acids. Section 5 discusses another real classification problem, where the objective is to detect Parkinson's disease. This example is provided to show how our method can be used not only for improving prediction accuracy, but also for identifying hidden structure in the data. Finally, Section 6 is devoted to discussion and future directions.

2. Methodology

We now describe our classification model, which we call dpMNL, in detail. We assume that for each case we observe a vector of continuous covariates, x, of dimension p. The response variable, y, is categorical, with J classes. To model the relationship between y and x, we non-parametrically model the joint distribution of y and x, in the form $P(x,y) = P(x)P(y|x)$, using a Dirichlet process mixture. Within each component of the mixture, the relationship between y and x (i.e., $P(y|x)$) is expressed using a linear function. The overall relationship becomes nonlinear if the mixture contains more than one component. This way, while we relax the assumption of linearity, the flexibility of the relationship is controlled.

Each component in the mixture model has parameters $\theta = (\mu, \sigma^2, \alpha, \beta)$. The distribution of x within a component is multivariate normal, with mean vector μ and diagonal covariance, with the vector σ^2 on the diagonal. The distribution of y given x within a component is given by a multinomial logit (MNL) model—for $j = 1, \ldots, J$,

$$P(y = j|x, \alpha, \beta) \quad = \quad \frac{\exp(\alpha_j + x^T \beta_j)}{\sum_{j'=1}^{J} \exp(\alpha_{j'} + x^T \beta_{j'})}.$$

The parameter α_j is scalar, and β_j is a vector of length p. Note that given x, the distribution of y does not depend on μ and σ. This representation of the MNL model is redundant, since one of the β_j's (where $j = 1, ..., J$) can be set to zero without changing the set of relationships expressible with the model, but removing this redundancy would make it difficult to specify a prior that treats all classes symmetrically. In this parameterization, what matters is the difference between the parameters of different classes.

In addition to the mixture view, which follows Equation (1) with $C \to \infty$, one can also view each observation, i, as having its own parameter, θ_i, drawn independently from a distribution drawn from a Dirichlet process, as in Equation (5):

$$\begin{aligned} \theta_i | G &\sim G, \text{ for } i = 1, \ldots, n, \\ G &\sim \mathcal{D}(G_0, \gamma). \end{aligned}$$

Since G will be discrete, many groups of observations will have identical θ_i, corresponding to components in the mixture view.

Although the covariates in each component are assumed to be independent with normal priors, this independence of covariates exists only locally (within a component). Their global (over all

components) dependency is modeled by assigning data to different components (i.e., clustering). The relationship between y and x within a component is captured using an MNL model. Therefore, the relationship is linear locally, but nonlinear globally.

We could assume that y and x are independent within components, and capture the dependence between the response and the covariates by clustering too. However, this may lead to poor performance (e.g., when predicting the response for new observations) if the dependence of y on x is difficult to capture using clustering alone. Alternatively, we could also assume that the covariates are dependent within a component. For continuous response variables, this becomes equivalent to the model proposed by Müller et al. (1996). If the covariates are in fact dependent, using full covariance matrices (as suggested by Müller et al., 1996) could result in a more parsimonious model since the number of mixture component would be smaller. However, as we discussed above, this approach may be practically infeasible for problems with a moderate to large number of covariates. We believe that our method is an appropriate compromise between these two alternatives.

We define G_0, which is a distribution over $\theta = (\mu, \sigma^2, \alpha, \beta)$, as follows:

$$\begin{aligned} \mu_l|\mu_0,\sigma_0 &\sim N(\mu_0,\sigma_0^2), \\ \log(\sigma_l^2)|M_\sigma,V_\sigma &\sim N(M_\sigma,V_\sigma^2), \\ \alpha_j|\tau &\sim N(0,\tau^2), \\ \beta_{jl}|\nu &\sim N(0,\nu^2). \end{aligned}$$

The parameters of G_0 may in turn depend on higher level hyperparameters. For example, we can regard the variances of coefficients as hyperparameters with the following priors:

$$\begin{aligned} \log(\tau^2)|M_\tau,V_\tau &\sim N(M_\tau,V_\tau^2), \\ \log(\nu^2)|M_\nu,V_\nu &\sim N(M_\nu,V_\nu^2). \end{aligned}$$

We use MCMC algorithms for posterior sampling. We could use Gibbs sampling if G_0 is the conjugate prior for the likelihood given by F. That is, we would repeatedly draw samples from $\theta_i|\theta_{-i},y_i$ (where $i = 1,...,n$) using the conditional distribution (6). Neal (2000) presented several algorithms for sampling from the posterior distribution of Dirichlet process mixtures when non-conjugate priors are used. Throughout this paper, we use Gibbs sampling with auxiliary parameters (Neal's algorithm 8).

This algorithm uses a Markov chain whose state consists of $c_1,...,c_n$ and $\phi = (\phi_c : c \in \{c_1,...,c_n\})$, so that $\theta_i = \phi_{c_i}$. In order to allow creation of new clusters, the algorithm temporarily supplements the ϕ_c parameters of existing clusters with m (or $m-1$) additional parameter values drawn from the prior, where m a postive integer that can be adjusted to give good performance. Each iteration of the Markov chain simulation operates as follows:

- For $i = 1,...,n$: Let k^- be the number of distinct c_j for $j \neq i$ and let $h = k^- + m$. Label these c_j with values in $\{1,...,k^-\}$. If $c_i = c_j$ for some $j \neq i$, draw values independently from G_0 for those ϕ_c for which $k^- < c \leq h$. If $c_i \neq c_j$ for all $j \neq i$, let c_i have the label $k^- + 1$, and draw values independently from G_0 for those ϕ_c where $k^- + 1 < c \leq h$. Draw a new value for c_i from $\{1,...,h\}$ using the following probabilities:

$$P(c_i = c|c_{-i},y_i,\phi_1,...,\phi_h) = \begin{cases} b\,\dfrac{n_{-i,c}}{n-1+\gamma}F(y_i,\phi_c) & \text{for } 1 \leq c \leq k^-, \\ \\ b\,\dfrac{\gamma/m}{n-1+\gamma}F(y_i,\phi_c) & \text{for } k^- < c \leq h, \end{cases}$$

where $n_{-i,c}$ is the number of c_j for $j \neq i$ that are equal to c, and b is the appropriate normalizing constant. Change the state to contain only those ϕ_c that are now associated with at least to one observation.

- For all $c \in \{c_1, ..., c_n\}$ draw a new value from the distribution $\phi_c \mid \{y_i \text{ such that } c_i = c\}$, or perform some update that leaves this distribution invariant.

Throughout this paper, we set $m = 5$. This algorithm resembles one proposed by MacEachern and Müller (1998), with the difference that the auxiliary parameters exist only temporarily, which avoids an inefficiency in MacEachern and Müller's algorithm.

Samples simulated from the posterior distribution are used to estimate posterior predictive probabilities. For a new case with covariates x', the posterior predictive probability of the response variable, y', is estimated as follows:

$$P(y' = j|x') = \frac{P(y' = j, x')}{P(x')},$$

where

$$\begin{aligned} P(y' = j, x') &= \frac{1}{S}\sum_{s=1}^{S} P(y' = j, x'|G_0, \theta^{(s)}), \\ P(x') &= \frac{1}{S}\sum_{s=1}^{S} P(x'|G_0, \theta^{(s)}). \end{aligned}$$

Here, S is the number of post-convergence samples from MCMC, and $\theta^{(s)}$ represents the set of parameters obtained at iteration s. Alternatively, we could predict new cases using $P(y' = j, x') = \frac{1}{S}\sum_{s=1}^{S} P(y' = j|x', G_0, \theta^{(s)})$. While this would be computationally faster, the above approach allows us to learn from the covariates of test cases when predicting their response values. Note also that the above predictive probabilities include the possibility that the test case is from a new cluster.

We use these posterior predictive probabilities to make predictions for test cases, by assigning each test case to the class with the highest posterior predictive probability. This is the optimal strategy for a simple 0/1 loss function. In general, we could use more problem-specific loss functions and modify our prediction strategy accordingly.

Implementations for all our models were coded in MATLAB, and are available online at `http://www.ics.uci.edu/~babaks/codes`.

3. Results for Simulated Data

In this section, we illustrate our dpMNL model using synthetic data, and compare it with other models as follows:

- A simple MNL model, fitted by maximum likelihood or Bayesian methods.
- A Bayesian MNL model with quadratic terms (i.e., $x_l x_k$, where $l = 1, ..., p$ and $k = 1, ..., p$), referred to as qMNL.
- A decision tree model (Breiman et al., 1993) that uses 10-fold cross-validation for pruning, as implemented by the MATLAB "treefit", "treetest" (for cross-validation) and "treeprune" functions.

- Support Vector Machines (SVMs) (Vapnik, 1995), implemented with the MATLAB "svmtrain" and "svmclassify" functions from the Bioinformatics toolbox. Both a linear SVM (LSVM) and a nonlinear SVM with radial basis function kernel (RBF-SVM) were tried.

When the number of classes in a classification problem was bigger than two, LSVM and RBF-SVM used the all-vs-all scheme as suggested by Allwein et al. (2000), Fürnkranz (2002), and Hsu and Lin (2002). In this scheme, $\binom{J}{2}$ binary classifiers are trained where each classifier separates a pair of classes. The predicted class for each test case is decided by using a majority voting scheme where the class with the highest number of votes among all binary classes wins. For RBF-SVM, the scaling parameter, λ, in the RBF kernel, $k(x,x') = \exp(-||x-x'||/2\lambda)$, was optimized based on a validation set comprised of 20% of training samples.

The models are compared with respect to their accuracy rate and the F_1 measure. Accuracy rate is defined as the percentage of the times the correct class is predicted. F_1 is a common measurement in machine learning defined as:

$$F_1 = \frac{1}{J}\sum_{j=1}^{J}\frac{2A_j}{2A_j+B_j+C_j},$$

where A_j is the number of cases which are correctly assigned to class j, B_j is the number cases incorrectly assigned to class j, and C_j is the number of cases which belong to the class j but are assigned to other classes.

We do two tests. In the first test, we generate data according to the dpMNL model. Our objective is to evaluate the performance of our model when the distribution of data is comprised of multiple components. In the second test, we generate data using a smooth nonlinear function. Our goal is to evaluate the robustness of our model when data actually come from a different model.

3.1 Simulation 1

The first test was on a synthetic four-way classification problem with five covariates. Data are generated according to our dpMNL model, except the number of components was fixed at two. Two hyperparameters defining G_0 were given the following priors:

$$\begin{aligned}\log(\tau^2) &\sim N(0,0.1^2),\\ \log(\nu^2) &\sim N(0,2^2).\end{aligned}$$

The prior for component parameters $\theta = (\mu,\sigma^2,\alpha,\beta)$ defined by this G_0 was

$$\begin{aligned}\mu_l &\sim N(0,1),\\ \log(\sigma_l^2) &\sim N(0,2^2),\\ \alpha_j|\tau &\sim N(0,\tau^2),\\ \beta_{jl}|\nu &\sim N(0,\nu^2),\end{aligned}$$

where $l = 1,...,5$ and $j = 1,...,4$. We randomly draw parameters θ_1 and θ_2 for two components as described from this prior. For each component, we then generate 5000 data points by first drawing $x_{il} \sim N(\mu_l,\sigma_l)$ and then sampling y using the following MNL model:

$$P(y=j|x,\alpha,\beta) = \frac{\exp(\alpha_j+x\beta_j)}{\sum_{j'=1}^{J}\exp(\alpha_{j'}+x\beta_{j'})}.$$

Model	Accuracy (%)	F_1 (%)
Baseline	45.57 (1.47)	15.48 (1.77)
MNL (Maximum Likelihood)	77.30 (1.23)	66.65 (1.41)
MNL	78.39 (1.32)	66.52 (1.72)
qMNL	83.60 (0.99)	74.16 (1.30)
Tree (Cross Validation)	70.87 (1.40)	55.82 (1.69)
LSVM	78.61 (1.17)	67.03 (1.51)
RBF-SVM	79.09 (0.99)	63.65 (1.44)
dpMNL	89.21 (0.65)	81.00 (1.23)

Table 1: Simulation 1: the average performance of models based on 50 simulated data sets. The Baseline model assigns test cases to the class with the highest frequency in the training set. Standard errors of estimates (based on 50 repetitions) are provided in parentheses.

The overall sample size is 10000. We randomly split the data into a training set, with 100 data points, and a test set, with 9900 data points. We use the training set to fit the models, and use the independent test set to evaluate their performance. The regression parameters of the Bayesian MNL model with Bayesian estimation and the qMNL model have the following priors:

$$\begin{aligned} \alpha_j|\tau &\sim N(0,\tau^2), \\ \beta_{jl}|\nu &\sim N(0,\nu^2), \\ \log(\tau) &\sim N(0,1^2), \\ \log(\nu) &\sim N(0,2^2). \end{aligned}$$

The above procedure was repeated 50 times. Each time, new hyperparameters, τ^2 and ν^2, and new component parameters, θ_1 and θ_2, were sampled, and a new data set was created based on these θ's.

We used Hamiltonian dynamics (Neal, 1993) for updating the regression parameters (the α's and β's). For all other parameters, we used single-variable slice sampling (Neal, 2003) with the "stepping out" procedure to find an interval around the current point, and then the "shrinkage" procedure to sample from this interval. We also used slice sampling for updating the concentration parameter γ, We used $\log(\gamma) \sim N(-3,2^2)$ as the prior, which, encourages smaller values of γ, and hence a smaller number of components. Note that the likelihood for γ depends only on C, the number of unique components (Neal, 2000; Escobar and West, 1995). For all models we ran 5000 MCMC iterations to sample from the posterior distributions. We discarded the initial 500 samples and used the rest for prediction.

Our dpMNL model has the highest computational cost compared to all other methods. Simulating the Markov chain took about 0.15 seconds per iteration using a MATLAB implementation on an UltraSPARC III machine (approximately 12.5 minutes for each simulated data set). Each MCMC iteration for the Bayesian MNL model took about 0.1 second (approximately 8 minutes for each data set). Training the RBF-SVM model (with optimization of the scale parameter) took approximately 1 second for each data set. Therefore, SVM models have a substantial advantage over our approach in terms of computational cost.

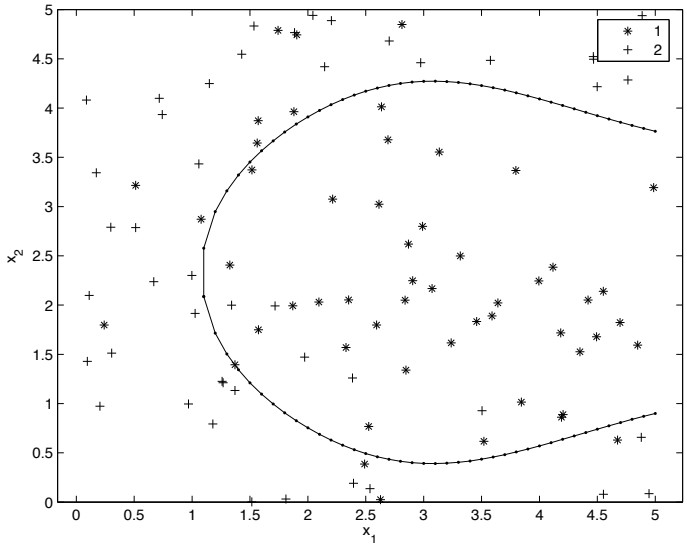


Figure 3: A random sample generated according to Simulation 2, with $a_3 = 0$. The dotted line is the optimal boundary function.

The average results (over 50 repetitions) are presented in Table 1. As we can see, our dpMNL model provides better results compared to all other models. The improvements are statistically significant (p-values < 0.001) for comparisons of accuracy rates using a paired t-test with $n = 50$.

3.2 Simulation 2

In the above simulation, since the data were generated according to the dpMNL model, it is not surprising that this model had the best performance compared to other models. In fact, as we increase the number of components, the amount of improvement using our model becomes more and more substantial (results not shown). To evaluate the robustness of the dpMNL model, we performed another test. This time, we generated x_{i1}, x_{i2}, x_{i3} (where $i = 1, ..., 10000$) from the $Uniform(0,5)$ distribution, and generated a binary response variable, y_i, according the following model:

$$P(y = 1|x) = \frac{1}{1 + \exp[a_1 \sin(x_1^{1.04} + 1.2) + x_1 \cos(a_2 x_2 + 0.7) + a_3 x_3 - 2]},$$

where a_1, a_2 and a_3 are randomly sampled from $N(1, 0.5^2)$. The function used to generate y is a smooth nonlinear function of covariates. The covariates are not clustered, so the generated data do not conform with the assumptions of our model. Moreover, this function includes a completely arbitrary set of constants to ensure the results are generalizable. Figure 3 shows a random sample from this model except that a_3 is fixed at zero (so x_3 is ignored). In this figure, the dotted line is the optimal decision boundary.

Table 2 shows the results for this simulation, which are averages over 50 data sets. For each data set, we generated 10000 cases by sampling new values for a_1, a_2, and a_3, new covariates, x, for each case, and new values for the response variable, y, in each case. As before, models were trained on 100 cases, and tested on the remaining 9900. As before, the dpMNL model provides significantly

Model	Accuracy (%)	F_1 (%)
Baseline	61.96 (1.53)	37.99 (0.57)
MNL (Maximum Likelihood)	73.58 (0.96)	68.33 (1.17)
MNL	73.58 (0.97)	67.92 (1.41)
qMNL	75.60 (0.98)	70.12 (1.36)
Tree (Cross Validation)	73.47 (0.95)	66.94 (1.43)
LSVM Linear	73.09 (0.99)	64.95 (1.71)
RBF-SVM	76.06 (0.94)	68.46 (1.77)
dpMNL	77.80 (0.86)	73.13 (1.26)

Table 2: Simulation 2: the average performance of models based on 50 simulated data sets. The Baseline model assigns test cases to the class with the highest frequency in the training set. Standard errors of estimates (based on 50 repetitions) are provided in parentheses.

(all p-values are smaller than 0.001) better performance compared to all other models. This time, however, the performances of qMNL and RBF-SVM are closer to the performance of the dpMNL model.

4. Results on Real Classification Problems

In this section, we first apply our model the problem of predicting a protein's 3D structure (i.e., folding class) based on its sequence. We then use our model to identify patients with Parkinson's disease (PD) based on their speech signals.

4.1 Protein Fold Classification

When predicting a protein's 3D structure, it is common to presume that the number of possible folds is fixed, and use a classification model to assign a protein to one of these folding classes. There are more than 600 folding patterns identified in the SCOP (Structural Classification of Proteins) database (Lo Conte et al., 2000). In this database, proteins are considered to have the same folding class if they have the same major secondary structure in the same arrangement with the same topological connections.

We apply our model to a protein fold recognition data set provided by Ding and Dubchak (2001). The proteins in this data set are obtained from the PDB_select database (Hobohm et al., 1992; Hobohm and Sander, 1994) such that two proteins have no more than 35% of the sequence identity for aligned subsequences larger than 80 residues. Originally, the resulting data set included 128 unique folds. However, Ding and Dubchak (2001) selected only the 27 most populated folds (311 proteins) for their analysis. They evaluated their models based on an independent sample (i.e., test set) obtained from PDB-40D (Lo Conte et al., 2000). PDB-40D contains the SCOP sequences with less than 40% identity with each other. Ding and Dubchak (2001) selected 383 representatives of the same 27 folds in the training set with no more than 35% identity to the training sequences. The training and test data sets are available online at `http://crd.lbl.gov/~cding/protein/`.

The covariates in these data sets are the length of the protein sequence, and the percentage composition of the 20 amino acids. While there might exist more informative covariates to predict protein folds, we use these so that we can compare the results of our model to that of Ding

and Dubchak (2001), who trained several Support Vector Machines (SVM) with nonlinear kernel functions.

We centered the covariates so they have mean zero, and used the following priors for the MNL model and qMNL model (with no interactions, only x_i and x_i^2 as covariates):

$$\begin{aligned}
\alpha_j|\eta &\sim N(0,\eta^2),\\
\log(\eta^2) &\sim N(0,2^2),\\
\beta_{jl}|\xi,\sigma_l &\sim N(0,\xi^2\sigma_l^2),\\
\log(\xi^2) &\sim N(0,1),\\
\log(\sigma_l^2) &\sim N(-3,4^2).
\end{aligned}$$

Here, the hyperparameters for the variances of regression parameters are more elaborate than in the previous section. One hyperparameter, σ_l, is used to control the variance of all coefficients, β_{jl} (where $j = 1,...,J$), for covariate x_l. If a covariate is irrelevant, its hyperparameter will tend to be small, forcing the coefficients for that covariate to be near zero. This method, termed Automatic Relevance Determination (ARD), has previously been applied to neural network models by Neal (1996). We also used another hyperparameter, ξ, to control the overall magnitude of all β's. This way, σ_l controls the relevance of covariate x_l compared to other covariates, and ξ controls the overall usefulness of all covariates in separating all classes. The standard deviation of β_{jl} is therefore equal to $\xi\sigma_l$.

The above scheme was also used for the dpMNL model. Note that in this model, one σ_l controls all β_{jlc}, where $j = 1,...,J$ indexes classes, and $c = 1,...,C$ indexes the unique components in the mixture. Therefore, the standard deviation of β_{jlc} is $\xi\sigma_l\nu_c$. Here, ν_c is specific to each component c, and controls the overall effect of coefficients in that component. That is, while σ and ξ are global hyperparameters common between all components, ν_c is a local hyperparameter within a component. Similarly, the standard deviation of intercepts, α_{jc} in component c is $\eta\tau_c$. We used $N(0,1)$ as the prior for ν_c and τ_c.

We also needed to specify priors for μ_l and σ_l, the mean and standard deviation of covariate x_l, where $l = 1,...,p$. For these parameters, we used the following priors:

$$\begin{aligned}
\mu_{lc}|\mu_{0,l},\sigma_{0,l} &\sim N(\mu_{0,l},\sigma_{0,l}^2),\\
\mu_{0,l} &\sim N(0,5^2),\\
\log(\sigma_{0,l}^2) &\sim N(0,2^2),\\
\log(\sigma_{lc}^2)|M_{\sigma,l},V_{\sigma,l} &\sim N(M_{\sigma,l},V_{\sigma,l}^2),\\
M_{\sigma,l} &\sim N(0,1^2),\\
\log(V_{\sigma,l}^2) &\sim N(0,2^2).
\end{aligned}$$

These priors make use of higher level hyperparameters to provide flexibility. For example, if the components are not different with respect to covariate x_l, the corresponding variance, $\sigma_{0,l}^2$, becomes small, forcing μ_{lc} close to their overall mean, $\mu_{0,l}$.

The MCMC chains for MNL, qMNL, and dpMNL ran for 10000 iterations. Simulating the Markov chain took about 2.1 seconds per iteration (5.8 hours in total) for dpMNL and 0.5 seconds per iteration (1.4 hours in total) for MNL using a MATLAB implementation on an UltraSPARC III machine. Training the RBF-SVM model took about 2.5 minutes.

Model	Accuracy (%)	F_1 (%)
MNL	50.0	41.2
qMNL	50.5	42.1
SVM (Ding and Dubchak, 2001)	49.4	-
LSVM	50.5	47.3
RBF-SVM	53.1	49.5
dpMNL	58.6	53.0

Table 3: Performance of models based on protein fold classification data.

The results for MNL, qMNL, LSVM, RBF-SVM, and dpMNL are presented in Table 3, along with the results for the best SVM model developed by Ding and Dubchak (2001) on the exact same data set. As we can see, the nonlinear RBF-SVM model that we fit has better accuracy than the linear models. Our dpMNL model provides an additional improvement over the RBF-SVM model. This shows that there is in fact a nonlinear relationship between folding classes and the composition of amino acids, and our nonlinear model could successfully identify this relationship.

4.2 Detecting Parkinson's Disease

The above example shows that our method can potentially improve prediction accuracy, though of course other classifiers, such as SVM and neural networks, may do better on some problems. However, we believe the application of our method is not limited to simply improving prediction accuracy—it can also be used to discover hidden structure in data by identifying subgroups (i.e., mixture components) in the population. This section provides an example to illustrate this concept.

Neurological disorders such as Parkinson's disease (PD) have profound consequences for patients, their families, and society. Although there is no cure for PD at this time, it is possible to alleviate its symptoms significantly, especially at the early stages of the disease (Singh et al., 2007). Since approximately 90% of patients exhibit some form of vocal impairment (Ho et al., 1998), and research has shown that vocal impairment could be one of the earliest indicators of onset of the illness (Duffy, 2005), voice measurement has been proposed as a reliable tool to detect and monitor PD (Sapir et al., 2007; Rahn et al., 2007; Little et al., 2008). For example, patients with PD commonly display a symptom known as *dysphonia*, which is an impairment in the normal production of vocal sounds.

In a recent paper, Little et al. (2008) show that by detecting *dysphonia*, we could identify patients with PD. Their study used data on sustained vowel phonations from 31 subjects, of whom 23 were diagnosed with PD. The 22 covariates used include traditional variables, such as measures of vocal fundamental frequency and measures of variation in amplitude of signals, as well as a novel measurement referred to as *pitch period entropy* (PPE). See Little et al. (2008) for a detailed description of these variables. This data set is publicly available at UCI Machine Learning Repository (`http://archive.ics.uci.edu/ml/datasets/Parkinsons`).

Little et al. (2008) use an SVM classifier with Gaussian radial basis kernel functions to identify patients with PD, chosing the SVM penalty value and kernel bandwidth by an exhaustive search over a range of values. They also perform an exhaustive search to select the optimal subset of features (10 features were selected). Their best model provides a 91.4% ($\pm$4.4) accuracy rate based on a bootstrap algorithm. This of course does not reflect the true prediction accuracy rate of the model for future observations since the model is trained and evaluated on the same sample. Here, we use

Model	Accuracy (%)	F_1 (%)
MNL	85.6 (2.2)	79.1 (2.8)
qMNL	86.1 (1.5)	79.7 (2.1)
LSVM	87.2 (2.3)	80.6 (2.8)
RBF-SVM	87.2 (2.7)	79.9 (3.2)
dpMNL	87.7 (3.3)	82.6 (2.5)

Table 4: Performance of models based on detecting Parkinson's disease. Standard errors of estimates (based on 5 cross-validation folds) are provided in parentheses.

Group	Frequency	Age Average	Male Proportion
1	107	66 (0.7)	0.86 (0.03)
2	12	72 (1.1)	0.83 (0.11)
3	36	63 (1.8)	0.08 (0.04)
4	40	65 (2.2)	0.40 (0.08)
Population	195	66 (0.7)	0.60 (0.03)

Table 5: The age average and male proportion for each cluster (i.e., mixture component) identified by our model. Standard errors of estimates are provided in parentheses.

a 5-fold cross validation scheme instead in order to obtain a more accurate estimate of prediction accuracy rate and avoid inflating model performance due to overfitting. As a result, our models cannot be directly compared to that of Little et al. (2008).

We apply our dpMNL model to the same data set, along with MNL and qMNL (no interactions, only x_i and x_i^2 as covariates). Although the observations from the same subject are not independent, we assume they are, as done by Little et al. (2008). Instead of selecting an optimum subset of features, we used PCA and chose the first 10 principal components. The MCMC algorithm for MNL, qMNL, and dpMNL ran for 3000 iterations (the first 500 iterations were discarded). Simulating the Markov chain took about 0.7 second per iteration (35 minutes per data set) for dpMNL and 0.1 second per iteration (5 minutes per data set) for MNL using a MATLAB implementation on an UltraSPARC III machine. Training the RBF-SVM model took 38 seconds for each data set.

Using the dpMNL model, the most probable number of components in the posterior is four (note that might change from one iteration to another). Table 4 shows the average and standard errors (based on 5-fold cross validation) of the accuracy rate and the F_1 measure for MNL, LSVM, RBF-SVM, and dpMNL. (But note that the standard errors assume independence of cross-validation folds, which is not really correct.)

While dpMNL provides slightly better results, the improvement is not statistically significant. However, examining the clusters (i.e., mixture components) identified by dpMNL reveals some information about the underlying structure in the data. Table 5 shows the average age of subjects and male proportion for the four clusters (based on the most probable number of components in the posterior) identified by our model. Note that age and gender are not available from the UCI Machine Learning Repository, and they are not included in our model. They are, however, available from Table 1 in Little et al. (2008). The first two groups include substantially higher percentages of male subjects than female subjects. The average age in the second of these groups is higher compared to the first group. Most of the subjects in the third group are female (only 8% are male).

The fourth group also includes more female subjects than male subjects, but the disproportionality is not as high as for the third group.

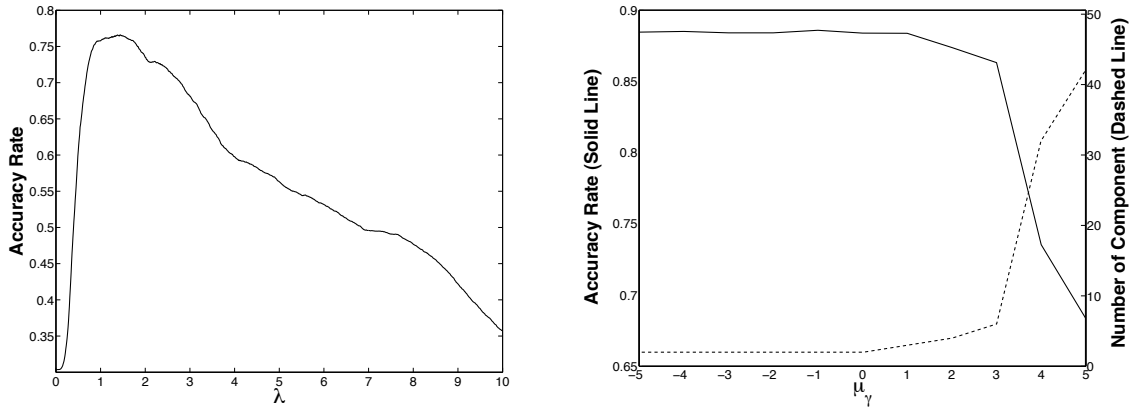
When identifying Parkinson's disease by detecting dysphonia, it has been shown that gender has a confounding effect (Cnockaert et al., 2008). By grouping the data into clusters, our model has identified (to some extent) the heterogeneity of subjects due to age and gender, even though these covariates were not available to the model. Moreover, by fitting a separate linear model to each component (i.e., conditioning on mixture identifiers), our model approximates the confounding effect of age and gender. For this example, we could have simply taken the age and gender of subjects from Table 1 in Little et al. (2008) and included them in our model. In many situations, however, not all the relevant factors are measured. This could result in unobservable changes in the structure of data. We discuss this concept in more detail elsewhere (Shahbaba, 2009).

5. Discussion and Future Directions

We introduced a new nonlinear classification model, which uses Dirichlet process mixtures to model the joint distribution of the response variable, y, and the covariates, x, non-parametrically. We compared our model to several linear and nonlinear alternative methods using both simulated and real data. We found that when the relationship between y and x is nonlinear, our approach provides substantial improvement over alternative methods. One advantage of this approach is that if the relationship is in fact linear, the model can easily reduce to a linear model by using only one component in the mixture. This way, it avoids overfitting, which is a common challenge in many nonlinear models.

We believe our model can provide more interpretable results. In many real problems, the identified components may correspond to a meaningful segmentation of data. Since the relationship between y and x remains linear in each segment, the results of our model can be expressed as a set of linear patterns for different data segments.

Hyperparameters such as λ in RBF-SVM and γ in dpMNL can substantially influence the performance of the models. Therefore, it is essential to choose these parameters appropriately. For RBF-SVM, we optimized λ using a validation set that includes 20% of the training data. Figure 4 (a) shows the effect of λ on prediction accuracy for one data set. The value of λ with the highest accuracy rate based on the validation set was used to train the RBF-SVM model. The hyperparameters in our dpMNL model are not fixed at some "optimum" values. Instead, we use hyperpriors that reflect our opinion regarding the possible values of these parameters before observing the data, with the posterior for these parameters reflecting both this prior opinion and the data. Hyperpriors for regression parameters, β, facilitate their shrinkage towards zero if they are not relevant to the classification task. The hyperprior for the scale parameter γ affects how many mixture components are present in the data. Instead of setting γ to some constant number, we allow the model to decide the appropriate value of γ, using a hyperprior that encourages a small number of components, but which is not very restrictive, and hence allows γ to become large in the posterior if required to fit the data. Choosing unreasonably restrictive priors could have a negative effect on model performance and MCMC convergence. Figure 4 (b) illustrates the negative effect of unreasonable priors for γ. For this data set, the correct number of components is two. We gradually increase μ_γ, where $\log(\gamma) \sim N(\mu_\gamma, 2)$, in order to put higher probability on larger values of γ and lower probability on smaller values. As we can see, setting $\mu_\gamma \geq 4$, which makes the hyperprior very restrictive, results in



(a) (b)

Figure 4: Effects of scale parameters when fitting a data set generated according to Simulation 1. (a) The effect of λ in the RBF-SVM model. (b) The effect of the prior on the scale parameter, $\gamma \sim \text{log-N}(\mu_\gamma, 2^2)$, as μ_γ changes in the dpMNL model.

a substantial decline in accuracy rate (solid line) due to overfitting with a large number of mixture components (dashed line).

The computational cost for our model is substantially higher compared to other methods such as MNL and SVM. This could be a preventive factor in applying our model to some problems. The computational cost of our model could be reduced by using more efficient methods, such as the "split-merge" approach introduced by Jain and Neal (2007). This method uses a Metropolis-Hastings procedure that resamples clusters of observations simultaneously rather than incrementally assigning one observation at a time to mixture components. Alternatively, it might be possible to reduce the computational cost by using a variational inference algorithm similar to the one proposed by Blei and Jordan (2005). In this approach, the posterior distribution P is approximated by a tractable variational distribution Q, whose free variational parameters are adjusted until a reasonable approximation to P is achieved.

We expect our model to outperform other nonlinear models such as neural networks and SVM (with nonlinear kernel functions) when the population is comprised of subgroups each with their own distinct pattern of relationship between covariates and response variable. We also believe that our model could perform well if the true function relating covariates to response variable contains sharp changes.

The performance of our model could be negatively affected if the covariates are highly correlated with each other. In such situations, the assumption of diagonal covariance matrix for x adopted by our model could be very restrictive. To capture the interdependencies between covariates, our model would attempt to increase the number of mixture components (i.e., clusters). This however is not very efficient. To address this issue, we could use mixtures of factor analyzers, where the covariance structure of high dimensional data is model using a small number of latent variables (see for example, Rubin and Thayer, 2007; Ghahramani and Hinton, 1996).

In this paper, we considered only continuous covariates. Our approach can be easily extended to situations where the covariate are categorical. For these problems, we need to replace the nor-

mal distribution in the baseline, G_0, with a more appropriate distribution. For example, when the covariate x is binary, we can assume $x \sim Bernoulli(\mu)$, and specify an appropriate prior distribution (e.g., *Beta* distribution) for μ. Alternatively, we can use a continuous latent variable, z, such that $\mu = \exp(z)/\{1+\exp(z)\}$. This way, we can still model the distribution of z as a mixture of normals. For categorical covariates, we can either use a Dirichlet prior for the probabilities of the K categories, or use K continuous latent variables, $z_1, ..., z_K$, and let the probability of category j be $\exp(z_j)/\sum_{j'}^{K}\exp(z_{j'})$.

Throughout this paper, we assumed that the relationship between y and x is linear within each component of the mixture. It is possible of course to relax this assumption in order to obtain more flexibility. For example, we can include some nonlinear transformation of the original variables (e.g., quadratic terms) in the model.

Our model can also be extended to problems where the response variable is not multinomial. For example, we can use this approach for regression problems with continuous response, y, which could be assumed normal within a component. We would model the mean of this normal distribution as a linear function of covariates for cases that belong to that component. Other types of response variables (i.e., with Poisson distribution) can be handled in a similar way.

In the protein fold prediction problem discussed in this paper, classes were regarded as a set of unrelated entities. However, these classes are not completely unrelated, and can be grouped into four major structural classes known as α, β, α/β, and $\alpha+\beta$. Ding and Dubchak (2001) show the corresponding hierarchical scheme (Table 1 in their paper). We have previously introduced a new approach for modeling hierarchical classes (Shahbaba and Neal, 2006, 2007). In this approach, we use a Bayesian form of the multinomial logit model, called corMNL, with a prior that introduces correlations between the parameters for classes that are nearby in the hierarchy. Our dpMNL model can be extend to classification problems where classes have a hierarchical structure (Shahbaba, 2007). For this purposse, we use a corMNL model, instead of MNL, to capture the relationship between the covariates, x, and the response variable, y, within each component. The results is a nonlinear model which takes the hierarchical structure of classes into account.

Finally, our approach provides a convenient framework for semi-supervised learning, in which both labeled and unlabeled data are used in the learning process. In our approach, unlabeled data can contribute to modeling the distribution of covariates, x, while only labeled data are used to identify the dependence between y and x. This is a quite useful approach for problems where the response variable is known for a limited number of cases, but a large amount of unlabeled data can be generated. One such problem is classification of web documents.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada. Radford Neal holds a Canada Research Chair in Statistics and Machine Learning.

References

E. L. Allwein, R. E. Schapire, Y. Singer, and P. Kaelbling. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

C. E. Antoniak. Mixture of Dirichlet process with applications to Bayesian nonparametric problems. *Annals of Statistics*, 273(5281):1152–1174, 1974.

D. Blackwell and J. B. MacQueen. Ferguson distributions via polya urn scheme. *Annals of Statistics*, 1:353–355, 1973.

D. M. Blei and M. I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1:121–144, 2005.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall, Boca Raton, 1993.

C. A. Bush and S. N. MacEachern. A semi-parametric Bayesian model for randomized block designs. *Biometrika*, 83:275–286, 1996.

B. Cai and D. B. Dunson. Bayesian covariance selection in generalized linear mixed models. *Biometrics*, 62:446–457, 2006.

L. Cnockaert, J. Schoentgen, P. Auzou, C. Ozsancak, L. Defebvre, and F. Grenez. Low-frequency vocal modulations in vowels produced by parkinsonian subjects. *Speech Communication*, 50(4): 288–300, 2008.

M. J. Daniels and R. E. Kass. Nonconjugate Bayesian estimation of covariance matrices and its use in hierarchical models. *Journal of the American Statistical Association*, 94(448):1254–1263, 1999.

C. H. Q. Ding and I. Dubchak. Multi-class protein fold recognision using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

J. R. Duffy. *Motor Speech Disorders: Substrates, Differential Diagnosis and Management*. Elsevier Mosby, St. Louis, Mo., 2nd edition, 2005.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of American Statistical Society*, 90:577–588, 1995.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1: 209–230, 1973.

T. S. Ferguson. *Recent Advances in Statistics, Ed: Rizvi, H. and Rustagi, J.*, chapter Bayesian density estimation by mixtures of normal distributions, pages 287–302. Academic Press, New York, 1983.

J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002. ISSN 1533-7928.

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

A. K. Ho, R. Iansek, C. Marigliani, J. L. Bradshaw, and S. Gates. Speech impairment in a large sample of patients with Parkinson's disease. *Behavioural Neurology*, 11:131–137, 1998.

U. Hobohm and C. Sander. Enlarged representative set of proteins. *Protein Science*, 3:522–524, 1994.

U. Hobohm, M. Scharf, R. Schneider, and C. Sander. Selection of a representative set of structure from the brookhaven protein bank. *Protein Science*, 1:409–417, 1992.

C. W. Hsu and C. J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

S. Jain and R. M. Neal. Splitting and merging components of a nonconjugate Dirichlet process mixture model (with discussion). *Bayesian Analysis*, 2:445–472, 2007.

M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman, and L. O. Ramig. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Transactions on Biomedical Engineering*, (in press), 2008.

L. Lo Conte, B. Ailey, T.J.P Hubbard, S. E. Brenner, A.G. Murzing, and C. Chothia. Scop: a structural classification of protein database. *Nucleic Acids Research*, 28:257–259, 2000.

S. N. MacEachern and P. Müller. Estimating mixture of Dirichlet process models. *Journal of Computational and Graphcal Statistics*, 7:223–238, 1998.

E. Meeds and S. Osindero. An alternative infinite mixture of Gaussian process experts. *Advances in Neural Information Processing Systems*, 18:883, 2006.

P. Müller, A. Erkanli, and M. West. Bayesian curve fitting using multivariate mormal mixtures. *Biometrika*, 83(1):67–79, 1996.

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:249–265, 2000.

R. M. Neal. Slice sampling. *Annals of Statistics*, 31(3):705–767, 2003.

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. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics No. 118, New York: Springer-Verlag, 1996.

D. A. Rahn, M. Chou, J. J. Jiang, and Y. Zhang. Phonatory impairment in Parkinson's disease: evidence from nonlinear dynamic analysis and perturbation analysis.(Clinical report). *Journal of Voice*, 21:64–71, 2007.

C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems 14*, page 881. MIT Press, 2002.

D. Rubin and D. Thayer. EM algorithms for ML factor analysis. *Pshychometrika*, 47(1):69–76, 2007.

S. Sapir, J. L. Spielman, L. O. Ramig, B. H. Story, and C. Fox. Effects of intensive voice treatment (the Lee Silverman Voice Treatment [lsvt]) on vowel articulation in dysarthric individuals with idiopathic Parkinson disease: Acoustic and perceptual findings. *Journal of Speech, Language, and Hearing Research*, 50(4):899–912, 2007.

B. Shahbaba. Discovering hidden structures using mixture models: Application to nonlinear time series processes. *Studies in Nonlinear Dynamics & Econometrics*, 13(2):Article 5, 2009.

B. Shahbaba. *Improving Classification Models When a Class Hierarchy is Available*. PhD thesis, Biostatistics, Public Health Sciences, University of Toronto, 2007.

B. Shahbaba and R. M. Neal. Improving classification when a class hierarchy is available using a hierarchy-based prior. *Bayesian Analysis*, 2(1):221–238, 2007.

B. Shahbaba and R. M. Neal. Gene function classification using Bayesian models with hierarchy-based priors. *BMC Bioinformatics*, 7:448, 2006.

N. Singh, V. Pillay, and Y. E. Choonara. Advances in the treatment of Parkinson's disease. *Progress in Neurobiology*, 81:29–44, 2007.

I. Ulusoy and C. M. Bishop. Generative versus discriminative methods for object recognition. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 258–265, Washington, DC, USA, 2005. IEEE Computer Society.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

S. Waterhouse, D. MacKay, and T. Robinson. Bayesian methods for mixtures of experts. In *Advances in Neural Information Processing Systems 8*, page 351. MIT Press, 1996.

Journal of Machine Learning Research 10 (2009) 1851-1880 Submitted 4/08; Revised 11/08; Published 8/09

CarpeDiem: Optimizing the Viterbi Algorithm and Applications to Supervised Sequential Learning

Roberto Esposito ESPOSITO@DI.UNITO.IT
Daniele P. Radicioni RADICION@DI.UNITO.IT
Department of Computer Science
University of Turin
Corso Svizzera, 185 - 10149 Turin - Italy

Editor: Michael Collins

Abstract

The growth of information available to learning systems and the increasing complexity of learning tasks determine the need for devising algorithms that scale well with respect to all learning parameters. In the context of supervised sequential learning, the Viterbi algorithm plays a fundamental role, by allowing the evaluation of the best (most probable) sequence of labels with a time complexity linear in the number of time events, and quadratic in the number of labels.

In this paper we propose CarpeDiem, a novel algorithm allowing the evaluation of the best possible sequence of labels with a sub-quadratic time complexity.[1] We provide theoretical grounding together with solid empirical results supporting two chief facts. CarpeDiem always finds the optimal solution requiring, in most cases, only a small fraction of the time taken by the Viterbi algorithm; meantime, CarpeDiem is never asymptotically worse than the Viterbi algorithm, thus confirming it as a sound replacement.

Keywords: Viterbi algorithm, sequence labeling, conditional models, classifiers optimization, exact inference

1. Introduction

In supervised learning systems, classifiers are learnt from sets of labeled examples and then used to predict the "correct" labeling for new objects. According to how relations between objects are exploited to build and evaluate the classifier, different categories of learning systems can be individuated. When the learning system deals with examples as isolated individuals, thus disregarding any relation among them, the system is said to work in a *propositional* setting. In this case classifiers can optimize the assignment of labels individually. Instead, in the setting of supervised sequential learning (SSL) the objects are assumed to be arranged in a sequence: relationships between previous and subsequent objects exist, and are used to improve the classification accuracy. SSL classifiers are then required to find the globally optimum sequence of labels, rather than the sequence of locally optimal labels. For instance, in the optical character recognition task, the labelling "*learning*" is probably better than "*learn1ng*", even though the description of the sixth character taken in isolation might suggest otherwise. A SSL classifier may deal with such ambiguities by exploiting the

1. The implementation of CarpeDiem and of several sequence learning algorithms can be downloaded at:
http://www.di.unito.it/~esposito/Software/seqlearning.tar.gz
a working GUI (Mac OS X only) for experimenting with the software can be downloaded at:
http://www.di.unito.it/~esposito/Software/SequenceLearningExperimenterBinaries.zip.

higher sequential correlation, in the English language, of the bigram *in* with respect to *1n*. Conceptually, given a sequence of T observations and K possible labels, K^T possible combinations of labels are to be considered by SSL classifiers. Most systems deal with such complexity by assuming that relations may span only over nearby objects and use the Viterbi algorithm (Viterbi, 1967) to find the globally optimal sequence of labels in $\Theta(TK^2)$ time.

In the last few years it has become increasingly important for supervised sequential learning algorithms to handle problems with large state spaces (Dietterich et al., 2008). Unfortunately, even the drastic reduction in complexity achieved by the Viterbi algorithm may be not sufficient in such domains. For instance, this is the case of web-logs related tasks (Felzenszwalb et al., 2003), music analysis (Radicioni and Esposito, 2007), and activity monitoring through body sensors (Siddiqi and Moore, 2005), where the number of possible labels is so large that the classification time can grow prohibitively high.

Some recent works propose techniques that under precise assumptions allow faster execution time of classifiers based on hidden Markov models (HMMs) (Rabiner, 1989). One feature shared by these approaches is the assumption that the transition matrix has a specific form allowing one to rule out most transitions. Such approaches are highly valuable when the problem naturally fits the assumption; *vice versa* they either lose the optimal solution or cannot be applied at all, when it does not. Moreover, they assume the transition matrix to be known beforehand and fixed over time. While this is a natural assumption in HMMs, recent algorithms based on the boolean features framework (McCallum et al., 2000) allow for more general settings where the transition matrix is itself a function of the observations around the object to be labelled. In such cases it is hard to figure out how the aforementioned approaches apply.

In this paper we introduce `CarpeDiem`. It is a parameter-free algorithm, sporting best case sub-quadratic complexity, devised as a replacement for the Viterbi algorithm. `CarpeDiem` avoids considering a transition whenever local observations make it impossible for the transition to be part of the optimal path. `CarpeDiem` preserves the optimality of the result, never being asymptotically worse than the Viterbi algorithm. Moreover, `CarpeDiem` automatically adapts to the sequence being evaluated, so that its complexity degrades to the Viterbi algorithm complexity in case the underlying assumption is not met. Interestingly, in contrast with alternative approaches, the assumption made by `CarpeDiem` needs not be "always" valid. On the contrary, the algorithm is able to take advantage of the assumption even when it holds for small portions of the sequence. This implies that the worst case complexity is hit only in the very unlikely situation where the assumption does not hold for the entire sequence. Finally, `CarpeDiem` can be directly applied in any sequential learning system based on the Viterbi algorithm, even in those where the transition matrix changes over time.

The present work is structured as follows: we briefly recall the Viterbi algorithm and state the problem (Section 2). After surveying related work (Section 3), we illustrate `CarpeDiem` in full detail, and an execution example on a toy problem is provided (Section 4). We then show how `CarpeDiem` can be embodied in the voted perceptron algorithm (Section 5) and, in Section 6, we report the experimental results and discuss the results as well as several related algorithms, and elaborate on future directions of research. The soundness of the algorithm as well as its complexity are formally proved in Appendices A and B, respectively.

2. Preliminaries

The problem of finding the best sequence of labels is often represented as a search for the optimal path in a layered and weighted graph (Figure 1).

Definition 1 Layered graph. *A layered graph is a connected graph where vertices are partitioned into a set of "layers" such that:* i*) edges connect only vertices in adjacent layers;* ii*) any vertex in a given layer is connected to all vertices of the successive layer.*

We adopt the convention of indicating the layer to which a vertex belongs as a subscript to the vertex name, so that y_t denotes a vertex in layer t. We associate to each vertex y_t a weight $S^0_{y_t}$, and to each edge (y_{t-1}, y_t) a weight $S^1_{y_t, y_{t-1}}$ (Figure 1). In the following we use the term "vertical" in referring to "per node" properties. For instance, we will use the expressions "vertical weight" of y_t and "vertical information" to refer to $S^0_{y_t}$ and to the information provided by evidence related to vertices, respectively. Similarly, we use the term "horizontal" in referring to "per edge" properties. For instance, we will use the expression "horizontal weight" in referring to the weight associated to a given transition. The distinction between vertical and horizontal information is important in the present work, the key idea in `CarpeDiem` is to exploit vertical information to avoid considering the horizontal one.

Given a layered and weighted graph with T layers and K vertices per layer, a *path* is a sequence of vertices $y_1, y_2, \ldots, y_t$ ($1 \leq t \leq T$). The reward for a path is the sum of the vertical and horizontal weights associated to the path:

$$\text{reward}(y_1, y_2, \ldots, y_t) = \left(\sum_{u=1}^{t-1} S^0_{y_u} + S^1_{y_{u+1}, y_u} \right) + S^0_{y_t}.$$

We define $\gamma(y_t)$ as the maximal reward associated to any path from any node in layer 1 to y_t:

$$\gamma(y_t) = \max_{y_1, y_2, \ldots, y_{t-1}} \text{reward}(y_1, y_2, \ldots, y_{t-1}, y_t).$$

We consider the problem of picking the maximal path from the leftmost layer to the rightmost layer. The *naive* solution considers all the K^T possible paths, and returns the maximal one. The Viterbi algorithm (Viterbi, 1967) solves the problem in $\Theta(TK^2)$ time by exploiting a dynamic programming strategy. The main idea stems from noticing that the reward of the best path to node y_t can be recursively computed as: *i)* the reward of the best path to the predecessor $\pi(y_t)$ on the optimal path to y_t; *ii)* plus the reward for transition $S^1_{y_t, \pi(y_t)}$; *iii)* plus the weight of node y_t. In formulae:

$$\gamma(y_t) = \begin{cases} S^0_{y_t} & \text{if } t = 1 \\ \gamma(\pi(y_t)) + S^1_{y_t, \pi(y_t)} + S^0_{y_t} & \text{otherwise.} \end{cases} \quad (1)$$

We will also make use of the equivalent formulation obtained by noticing that $\pi(y_t)$ is the best predecessor for y_t. That is, $\pi(y_t)$ is the vertex y_{t-1} (in layer $t-1$) that maximizes the quantity $\gamma(y_{t-1}) + S^1_{y_t, y_{t-1}}$. Then:

$$\gamma(y_t) = \begin{cases} S^0_{y_t} & \text{if } t = 1 \\ \max_{y_{t-1}} \left(\gamma(y_{t-1}) + S^1_{y_t, y_{t-1}} + S^0_{y_t} \right) & \text{otherwise.} \end{cases} \quad (2)$$

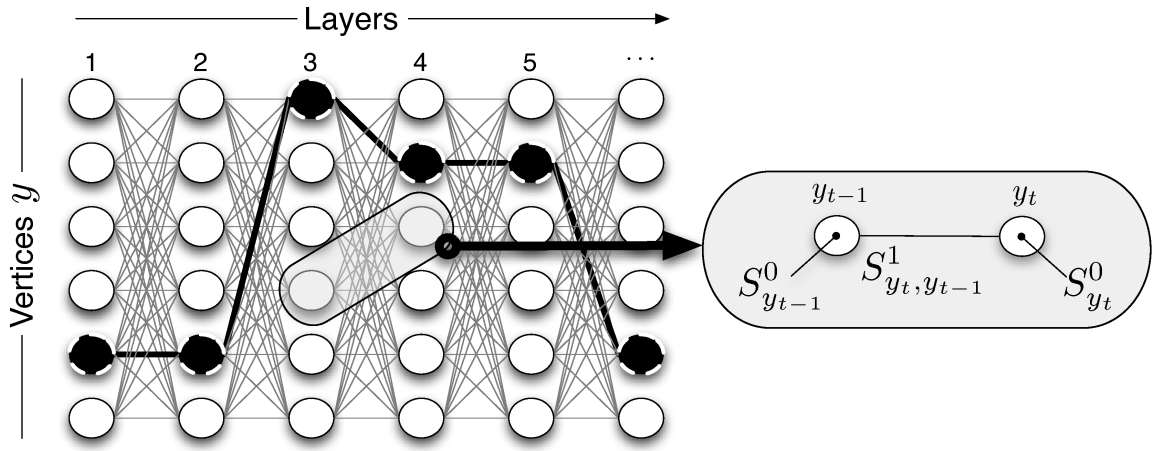


Figure 1: $S^0_{y_t}$ and $S^0_{y_{t-1}}$ denote per vertex (vertical) weights. $S^1_{y_t,y_{t-1}}$ denotes per edge (horizontal) weights.

The Viterbi algorithm proceeds from left to right storing the values of γ into an array $\mathbb{G}$ as soon as such values are computed. Assuming that $\forall y_{t-1} : \mathbb{G}(y_{t-1}) = \gamma(y_{t-1})$, then $\mathbb{G}(y_t)$ is computed as:

$$\mathbb{G}(y_t) = \max_{y_{t-1}} \left(\mathbb{G}(y_{t-1}) + S^1_{y_t,y_{t-1}} + S^0_{yt} \right).$$

The pseudo code for the algorithm is reported in Algorithm 1. Since the maximization at the line marked with label number 1 (henceforth simply "line 1 ") requires $\Theta(K)$ time, the time needed for processing each layer is in the order of $\Theta(K^2)$. The total time required by Viterbi is then $\Theta(K^2T)$. The standard formulation of the Viterbi algorithm would also store the optimal path information as it becomes available. Since this can be done using standard techniques (Cormen et al., 1990, page 520) without affecting the complexity of the algorithm, we do not explicitly report that in the pseudo-code.

Let us now consider how the above definitions instantiate in the context of a learning environment. The symbol $S^0_{y_t}$ conveys information about label y_t provided by observing the data at time t. In hidden Markov models terminology, $S^0_{y_t}$ corresponds to probability $b_{y_t}(x_t)$ of observing symbol x_t in state y_t (Rabiner, 1989, definition of $b_j(k)$ pag. 261, Eq. 8). More generally, $S^0_{y_t}$ is a quantity that depends on both the label y_t predicted for time (layer) t and the observations at and *around* time t. Likewise, in HMMs terminology, $S^1_{y',y}$ corresponds to the probability $a_{yy'}$ of transiting from state y to state y' (Rabiner, 1989, definition of a_{ij} pag. 260, Eq. 7). More in general, $S^1_{y',y}$ may depend on both the labels (y',y) *and* on the observations at and around the current layer. We note that since $S^1_{y',y}$ may vary over time (which motivates the notation $S^1_{y_t,y_{t-1}}$), the setup considered here is more general than the one of HMMs, where the transition matrix does not depend on the time instant.

```
begin
    forall y_1 do
        G(y_1) ← S^0_{y_1};
    end
    for t = 2 to T do
        forall y_t do
1           G(y_t) ← max_{y_{t-1}} ( G(y_{t-1}) + S^1_{y_t,y_{t-1}} + S^0_{y_t} );
        end
    end
    y*_T ← arg max_{y_T} G(y_T);
    return y*_T;
end
```

Algorithm 1: The Viterbi algorithm.

3. Related Work

As we illustrated in Section 1, in cases where hundreds or thousands of labels are to be handled, the quadratic dependence on the number of labels is still a high burden that limits the applicability of sequential learning techniques.

In other fields (e.g., telecommunications) there exist *ad hoc* solutions that allow one to tame the complexity of the Viterbi algorithm by means of hardware implementations (Austin et al., 1990) or methods for approximating the optimum path (Fano, 1963). For instance, in the research field of speech recognition, the Viterbi algorithm is routinely applied to huge problems. This is a typical case where approximate solutions really pay off: suboptimal paths could be tolerated (to some extent) and tight time constraints prevent exhaustive search. A popular approach in this field is the *Viterbi beam search* (VBS) (Lowerre and Reddy, 1980; Spohrer et al., 1980; Bridle et al., 1982): essentially, VBS performs a breadth-first suboptimal search in which only the most promising solutions are retained at each step. Many improvements over this basic strategy have been proposed to refine either the computational performance or the accuracy of the solution (e.g., Ney et al., 1992). In most cases domain-based knowledge (such as language constraints) is used to restrict the search efforts to some relevant regions of the search space (Ney et al., 1987). Also, in recent years, several algorithms have been proposed that overcome the difficulties inherent in heuristic ranking strategies by learning ranking functions specifically optimized for the problem at hand (Xu and Fern, 2007).

Although promising, the VBS approach does not come without difficulties. For instance, Collins and Roark (2004) propose Viterbi beam search to improve the performances of the perceptron algorithm on the particular problem of natural language parsing. Interestingly, the authors note how the sub-optimality of the beam search can negatively affect the learning performances. The problem arises when a sub-optimal sequence is used instead of the optimal one to update the weights of the features (please refer to Section 5). In order to alleviate this issue, the authors stop the search—during learning—as soon as the beam does not contain the optimal solution. In such case only the partial sequence, up to when the stopping occurred, is used to update the weights. This prevents from training the perceptron using "bad" predictions, but it still has the drawback of exploiting only partially the training sequences. In such system, then, the sub-optimality of Viterbi beam search

has *two* drawbacks: at learning time, it hinders the process of finding better classifiers (or at least it slows the process down); at testing time, it yields sub-optimal classifications.

In recent years, despite a widespread usage of the Viterbi algorithm within the sequential learning field, only few works addressed the problem of reducing its time complexity and at the same time retaining the optimal result. In Felzenszwalb et al. (2003), linear (and near linear) algorithms are proposed to compute the optimal labels sequence. Their algorithms work under the assumption that the reward for the transition between states number i and j is a "simple" function of $|i-j|$.[2]

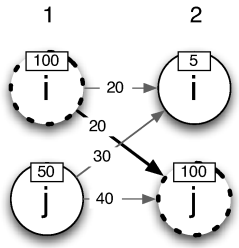
In Siddiqi and Moore (2005) it is assumed that the transition matrix is well approximated by a particular one where, for each vertex, the probability mass is concentrated on the k highest transitions leaving it. Then, the weights associated to the other transitions are approximated by a constant, and the optimal path is evaluated with $\Theta(kKT)$ time complexity. Clearly, the smaller k, the faster the algorithm.

Both techniques provide significant time savings with respect to the Viterbi algorithm. However, they are both based on assumptions about the entries in the transition matrix that are not guaranteed to hold in practice. More in particular, the assumption by Felzenszwalb et al. (2003) does not seem to easily fit general cases. Also, the investigation needed to devise the correct parameter space may require knowledge and efforts that are not always at disposal of the average practitioner. The assumption underlying the work of Siddiqi and Moore (2005) is, in our opinion, simpler to be fulfilled in practice. However, the extent to which it holds (which determines the magnitude of k) cannot be easily forecasted. Again, the extra efforts needed to assess the applicability of the approach may be detrimental to its application. Moreover, both approaches require *homogeneous* transition matrices: that is, transition matrices that do not vary over time. This is a common assumption, but unfortunately it cannot be guaranteed in some recently developed approaches, as those based on the boolean feature framework (McCallum et al., 2000). `CarpeDiem` can be safely applied even in this more complex scenario. In the experimentation, we successfully apply `CarpeDiem` in both settings, the one where the transition matrix is not constant (Section 6.1), as well as the one where it is (Sections 6.2 and 6.3).

In a recent paper Mozes et al. (2007) propose an *exact* compression-based technique to speed up the Viterbi algorithm. The authors propose to use three well known compression schemes achieving significant speed-ups whose magnitude depends on which compression algorithm is adopted. Interestingly the cited approach is not a search scheme, rather it is a preprocessing step. As such, it qualifies as an orthogonal technique amenable to be used together with `CarpeDiem` obtaining the advantages of both techniques.

In facts, to the best of our knowledge, the algorithms `CarpeDiem` and (Mozes et al., 2007) are the only exact ones, capable of speeding up the Viterbi algorithm when the assumption of homogeneous matrices is dropped. We would also argue that the other approaches presented above are not easily adapted to work in this, more complex, scenario. In Siddiqi and Moore (2005) the transition matrix needs to be traversed in advance in order to obtain the highest ranking frequencies. If those frequencies change over time, this operation needs to be repeated for each t, and the algorithm would require $O(TK^2)$ only to compute this preprocessing step. In Felzenszwalb et al. (2003) it is necessary to express the weights of the transition from label i to label j in terms of a function of $|i-j|$. The effectiveness of the approach depends on particular properties of this function. It could be argued that, in very particular situations, those properties could be shown to hold even when

2. One whose maximum can be calculated in (nearly) constant time.



Vertical weights	$S^0_{i_1}$	100	$S^0_{j_1}$	50	$S^0_{i_2}$	5	$S^0_{j_2}$	100
Horizontal weights	$S^1_{i_1,i_2}$	20	$S^1_{i_1,j_2}$	20	$S^1_{j_1,i_2}$	30	$S^1_{j_1,j_2}$	40

Figure 2: Sometimes horizontal weights can be disregarded without loosing the optimal path.

the transition matrix varies with t. However, it is hard to figure out a general way to enforce this property without inspecting the whole transition matrix at each time step.

`CarpeDiem` enjoys the desirable property of smoothly scaling to the Viterbi algorithm complexity when the underlying assumptions soften (as argued in Section 4.5). This has two consequences: 1) the algorithm adapts to the problem *and* to the sequence at hand, and 2) the algorithm can be universally applied (even when one is unsure about whether the problem fits `CarpeDiem` assumptions or not). In contrast with other state-of-the-art algorithms, no domain knowledge is to be given, nor any parameter needs to be set. This makes `CarpeDiem` well suited to be used on a regular basis as a *drop-in* replacement of the Viterbi algorithm: in the worst case, with no time saving.

4. The `CarpeDiem` Algorithm

In the general case, in order to determine the end point of the best path to a given layer, one can avoid inspecting all vertices in that layer. In particular, after sorting the vertices in layer t according to their vertical weight, the search can be stopped when the difference in vertical weight of the best node so far and the next vertex in the ordering is big enough to counterbalance any advantage that can be possibly derived from exploiting a better transition and/or a better ancestor.

To clarify this point, it is interesting to consider the minimal example reported in Figure 2. Let us assume that the reward for the maximal weight for any transition is 60. Our objective is to find the endpoint of the best path to each layer. For layer 1 we have no incoming paths, the best endpoint is simply the vertex with the maximal vertical weight: in the example, i_1. In our approach, we consider vertices with highest vertical weight first. Hence we start by calculating the reward of the optimal path to node j_2: in the example, the path i_1, j_2 (with score $S^0_{i_1} + S^1_{j_2,i_1} + S^0_{j_2} = 100 + 20 + 100 = 220$). We notice that the reward attainable by reaching i_2 cannot be higher than 165, computed as the sum of the reward for the best path to layer 1 (i.e., 100), plus the maximal weight for any transition (i.e., 60), plus the vertical weight of i_2 (i.e., 5). Therefore the endpoint of the best path to layer 2 must be j_2 and it is not necessary to calculate the reward for reaching i_2. In the course of the algorithm (hopefully) many vertices will be left unexplored by means of the above strategy. We note, however, that this does not prevent from the need of exploring those vertices in the following steps. When necessity arises `CarpeDiem` goes back through the previous layers gathering the required information. This is why `CarpeDiem` makes use of two procedures: one that finds the best vertex in

each layer (Algorithm 3), and one that finds the reward for reaching a given node by traversing the graph from right to left (Algorithm 4).

We say that a vertex is *open* if the reward of its best incoming path has been computed; otherwise the vertex is said to be *closed*. `CarpeDiem` finds the best vertex for each layer by calling Algorithm 3 (also referred to as *forward search strategy*), which leaves closed as many vertices as possible. The *backward search strategy* is called to open vertices whenever necessary.

The main procedure of `CarpeDiem` is presented in Algorithm 2; the forward and the backward search strategies are presented in Algorithms 3 and 4, respectively. Before detailing the algorithm, we need to introduce several definitions.

Definition 2 *Let us define:*

S^{1*} *: an upper bound to the maximal transition weight in the current graph*

$$S^{1*} \geq \max_{y_t, y_{t-1}} S^1_{y_t, y_{t-1}}; \tag{3}$$

γ^*_t *: the reward of the best path to any vertex in layer t (including the vertical weight of the ending vertex)*

$$\gamma^*_t = \max_{y_t} \gamma(y_t);$$

β_t *: an upper bound to the reward that can be obtained in reaching layer t* ($2 \leq t \leq T$)

$$\beta_t = \gamma^*_{t-1} + S^{1*}; \tag{4}$$

$\sqsupseteq_t$ *: a total ordering—based on vertical weights—of vertices at layer t.*

$$\sqsupseteq_t \equiv \{(y_t, y'_t) | S^0_{y_t} \geqslant S^0_{y'_t}\}. \tag{5}$$

Also, we say that vertex y_t is more promising *than vertex y'_t iff $y_t \sqsupseteq_t y'_t$.*

During execution, `CarpeDiem` calculates several values that are strictly connected to the definitions above. In particular $\mathbb{G}$ is a vector of $K \times T$ elements. $\mathbb{G}(y_t)$ contains the value of $\gamma(y_t)$ as calculated by `CarpeDiem`. Also, $\mathbb{B}$ is a vector of T elements. $\mathbb{B}_t$ contains the value of β_t as calculated by `CarpeDiem`. To a good extent, proving `CarpeDiem` correct will involve proving that, indeed, $\mathbb{G}(y_t) = \gamma(y_t)$ and $\mathbb{B}_t = \beta_t$.

4.1 Algorithm 2 – Main Procedure

Algorithm 2 initializes $\mathbb{G}(y_1)$ and $\mathbb{B}_2$ values and calls Algorithm 3 on all layers $2 \ldots T$. More specifically, $\mathbb{G}(y_1)$ is set to S^0_1 (as required by Equation 1). Also $\mathbb{B}_2$ is set to the maximal vertical weight found plus S^{1*} (as required by Equation 4).

4.2 Algorithm 3 – Forward search strategy

The forward strategy searches for the best vertex for layer t stopping as soon as this vertex can be determined unambiguously.

At the beginning of the analysis of each layer all vertices in the layer are *closed*. The algorithm scans vertices in the order given by $\sqsupseteq_t$. As mentioned at the beginning of Section 4, in the general

begin
  foreach y_1 { Initialization Step } **do**
2 $\mathbb{G}(y_1) \leftarrow S^0_{y_1}$; { Opens vertex y_1 }
  end
3 $y^*_1 \leftarrow \arg\max_{y_1}(\mathbb{G}(y_1))$;
  $\mathbb{B}_2 \leftarrow \mathbb{G}(y^*_1) + S^{1*}$;
  foreach *layer* $t \in 2 \dots T$ **do**
    $y^*_t \leftarrow$ result of Algorithm 3 on layer t;
  end
  return path to y^*_T;
end

Algorithm 2: `CarpeDiem`.

begin
  $y^*_t \leftarrow$ most promising vertex;
  $y'_t \leftarrow$ next vertex in the $\sqsupseteq_t$ ordering;
  Open vertex y^*_t {call Algorithm 4};
  while $\mathbb{G}(y^*_t) < \mathbb{B}_t + S^0_{y'_t}$ **do**
4   Open vertex y'_t {call Algorithm 4};
5    $y^*_t \leftarrow \arg\max_{y'' \in \{y^*_t, y'_t\}} [\mathbb{G}(y'')]$;
    $y'_t \leftarrow$ next vertex in the $\sqsupseteq_t$ ordering;
  end
6 $\mathbb{B}_{t+1} \leftarrow \mathbb{G}(y^*_t) + S^{1*}$;
  return y^*_t;
end

Algorithm 3: Forward search strategy.

case, the algorithm can avoid opening all vertices in every layer. The search is stopped when the difference in vertical weight of y^*_t and y'_t is big enough to counterbalance any advantage that can be possibly derived from exploiting a better transition and/or a better ancestor. In formulae, let $\pi(y^*_t)$ be the best predecessor for y^*_t, the (forward) search is stopped when the currently best vertex y^*_t and the next vertex y'_t in the $\sqsupseteq_t$ ordering satisfy:

$$\overbrace{S^0_{y^*_t} - S^0_{y'_t}}^{\substack{\text{accounts for a better vertical} \\ \text{weight of } y^*_t \text{ w.r.t. } y_t}} \geq \overbrace{\left(\gamma^*_{t-1} - \gamma(\pi(y^*_t))\right)}^{\substack{\text{accounts for a possibly bet-} \\ \text{ter predecessor of } y'_t \text{ w.r.t. } y^*_t}} + \overbrace{\left(S^{1*} - S^1_{y^*_t, \pi(y^*_t)}\right)}^{\substack{\text{accounts for a possibly bet-} \\ \text{ter transition from } y'_t \text{ prede-} \\ \text{cessor}}} . \tag{6}$$

The above formula is a direct consequence of the exit condition of the *while* loop of Algorithm 3, and it can be obtained by substituting[3] $\mathbb{B}$ and $\mathbb{G}$ with β and γ, and then expanding β and γ using their definitions (we repeat the relevant definitions in Table 1-*a* and *b*).

3. The soundness of the substitution is guaranteed by Theorems 1 and 2.

a)	Definition of β	$\beta_t = \gamma^*_{t-1} + S^{1*}$
b)	Definition of γ (see Eq. 1)	$\gamma(y^*_t) = \gamma(\pi(y^*_t)) + S^1_{y^*_t,\pi(y^*_t)} + S^0_{y^*_t}$

Table 1: Summary of few useful quantities

In case the stop criterion is not met, the algorithm calls Algorithm 4 (also referred to as the *backward strategy*) which sets $\mathbb{G}(y'_t) = \gamma(y'_t)$. If necessary, the "maximal" vertex y^*_t (the vertex that, so far, has associated maximal reward) is updated. Before exiting, $\mathbb{B}_{t+1}$ is readied for later use, and the best vertex is returned.

4.3 Algorithm 4 – Backward search strategy

The backward search strategy opens a vertex y_t by finding its best ancestor and setting $\mathbb{G}(y_t)$ accordingly. In much the same spirit as in the forward strategy, the algorithm saves some computation *i*) by exploiting $\sqsupseteq_{t-1}$ in order to inspect first the most promising vertices, and *ii*) by taking advantage of β_{t-1} in order to stop the search as soon as possible.

Data: A vertex y_t to be opened
begin
  $y^*_{t-1} \leftarrow$ most promising vertex;
  $y'_{t-1} \leftarrow$ next vertex in the $\sqsupseteq_{t-1}$ ordering;
  while y'_{t-1} *is open* **do**
    $y^*_{t-1} \leftarrow \arg\max_{y'' \in \{y'_{t-1}, y^*_{t-1}\}} \left[\mathbb{G}(y'') + S^1_{y_t,y''}\right]$;
    $y'_{t-1} \leftarrow$ next vertex in the $\sqsupseteq_{t-1}$ ordering;
  end
  while $\left(\mathbb{G}(y^*_{t-1}) + S^1_{y_t,y^*_{t-1}} < \mathbb{B}_{t-1} + S^0_{y'_{t-1}} + S^{1*}\right)$ **do**
    Open y'_{t-1} {call Algorithm 4};
    $y^*_{t-1} \leftarrow \arg\max_{y'' \in \{y'_{t-1}, y^*_{t-1}\}} \left[\mathbb{G}(y'') + S^1_{y_t,y''}\right]$;
    $y'_{t-1} \leftarrow$ next vertex in the $\sqsupseteq_{t-1}$ ordering;
  end
7 $\mathbb{G}(y_t) \leftarrow \mathbb{G}(y^*_{t-1}) + S^1_{y_t,y^*_{t-1}} + S^0_{y_t}$;
end

Algorithm 4: Backward search strategy to open y_t.

The first loop finds the best predecessor among the open vertices of layer $t-1$. In the second loop, we exploit the same idea behind the forward strategy. Let us inspect the exit condition of the second loop:

$$\mathbb{G}(y^*_{t-1}) + \mathbf{S^1_{y_t,y^*_{t-1}}} < \mathbb{B}_{t-1} + S^0_{y'_{t-1}} + \mathbf{S^{1*}}.$$

With the exception of the symbols in bold font, the formula is the same as the one in the exit condition of the while loop in Algorithm 3. The bold symbols take into account the transition to the target vertex. Namely, $\mathbf{S^1_{y_t,y^*_{t-1}}}$ takes into account the transition from the current best vertex (y^*_{t-1})

to the target vertex y_t and $\mathbf{S^{1*}}$ accounts for the maximal reward that a transition from y'_{t-1} to y_t can possibly obtain.

Also the internal working of the second loop is very similar to the one in the forward strategy. After opening (through a recursive call) y'_{t-1}, the current best vertex is set to the best of y'_{t-1} and y^*_{t-1}.

4.4 Example

In the following we provide a description of an execution of `CarpeDiem` over a toy problem. The problem consists of labeling a sequence containing four events and two labels (named *i* and *j*). The example is reported in Figure 3. The weight shown on the edge between labels y_{t-1} and y_t corresponds to $S^1_{y_t,y_{t-1}}$. The bound S^{1*} on the maximum horizontal reward is 60. Two further quantities are reported in the figure, and shown graphically by means of boxes placed on vertices: within rectangular boxes, we report the vertical weight of the vertex. Within rounded boxes, we report:

- $\mathbb{G}(y_t)$, if y_t is open;
- $\mathbb{B}_t + S^0_{y_t}$, if y_t is closed and $\mathbb{B}_t$ has already been computed;
- 0, otherwise.

Here we give a detailed description of the algorithm execution over the given graph.

step (a) At the beginning of the execution, all vertices are closed. The initialization steps in Algorithm 2 open all vertices in layer 1. Clearly, there is no reward for arriving at vertices in layer 0 and no incoming transitions to be taken into account. The best vertex in layer 1 is thus the vertex having the maximum vertical weight.

step (b) The analysis of layer 2 starts by opening the most promising vertex in that layer (vertex *j*). Since all vertices at layer 1 are open, the backward strategy already has complete information at disposal, and it does not need to enter the second loop to open j_2. Once $\mathbb{G}(j_2)$ has been computed, the algorithm compares this value to the bound on the weight of the best path to i_2. Since $S^0_{i_2} + \mathbb{B}_2 = 165$ cannot outperform $\mathbb{G}(j_2) = 220$, there is no need to open vertex i_2.

step (c) To open i_3, the backward strategy goes back to layer 2 and searches for the best path to that vertex. Again, vertex i_2 can be left closed, since there is no chance that the best path to i_3 traverses it. In fact,

$$\mathbb{B}_2 + S^0_{i_2} + S^{1*} = (100+60) + 5 + 60 = 225$$

cannot outperform the reward

$$\mathbb{G}(j_2) + S^1_{i_3,j_2} = 220 + 15 = 235$$

obtained by passing through j_2. Then $\mathbb{G}(i_3)$ is set to $235 + 100 = 335$.

Unfortunately, this does not allow to make a definitive decision about whether this is the best vertex of layer 3, since $\mathbb{B}_3 + S^0_{j_3}$ is $(220+60)+70 = 350$. Next step will thereby settle the question by opening vertex j_3.

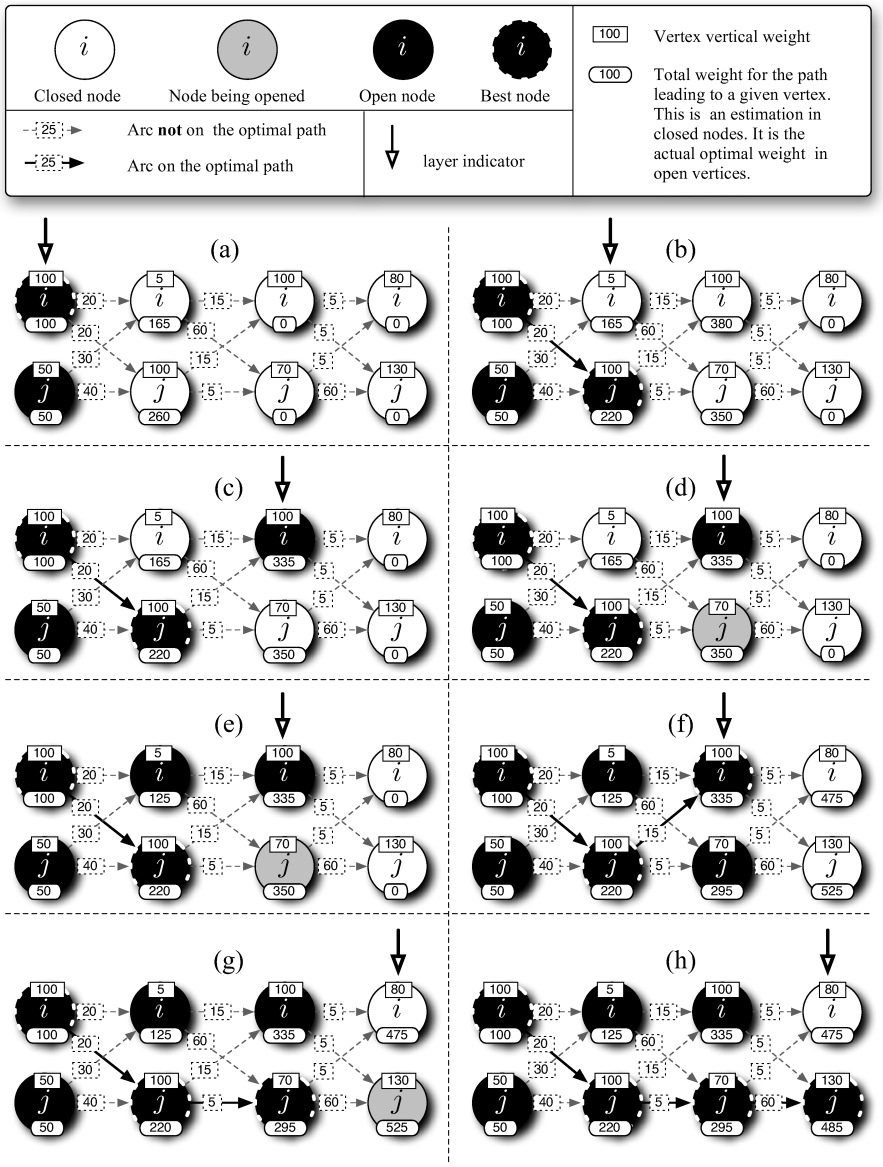


Figure 3: `CarpeDiem` in action on a toy problem.

step (d) The goal is, at this point, to find the best path to j_3. Even though j_2 has a clear advantage over i_2, this does not suffice to exclude that the latter one is on the optimal path to j_3 (since $\mathbb{G}(j_2) + S^1_{j_3,j_2} \not< \mathbb{B}_2 + S^0_{i_2} + S^{1*}$): the backward strategy is then forced to recursively call itself to open i_2.

step (e) By opening i_2, the algorithm sets $\mathbb{G}(i_2)$ to 125 (the best path being $i_1 \rightarrow i_2$), thus ruling it out as a candidate for being on the optimal path to j_3.

step (f) In returning to consider layer 3, we are back to the path $i_1 \rightarrow j_2 \rightarrow i_3$. To open vertices i_2 and j_3 has been wasteful, though unavoidable.

step (g) The first vertex to be opened in its layer is j_4. Interestingly, the best path to j_4 is not through the best vertex in layer 3. In fact, while the highest reward for a three steps walk is on vertex i_3, it is more convenient to go through vertex j_3 to reach vertex j_4.

step (h) Since $\mathbb{G}(j_4) = 485$ is larger than $\mathbb{B}_4 + S^0_{i_4}$, the algorithm terminates leaving i_4 closed.

In Section 6 real world problems are considered, and much larger optimizations obtained.

4.5 Algorithm Properties

An intuitive way of characterizing the algorithm complexity is to consider Formula 6, where the exit condition of Algorithm 3 is rewritten in order to point out why in some cases it is safe to stop inspecting the current layer. Clearly, the sooner the loop exit condition is satisfied, the faster the algorithm.

Arguably, the worst case happens when vertical rewards, being equal for each label, do not provide any discriminative power. In such a case, the left term in Formula 6 is zero, the inequality is never satisfied, and Algorithm 3 calls Algorithm 4 over all K vertices in every layer. In this case, for each one of the k vertices to be opened in a new layer, the first loop of Algorithm 4 iterates over all K predecessors. However, no recursive call takes place. Overall, in the worst case hypothesis, `CarpeDiem` has order of $O(TKK + TK\log(K)) = O(TK^2)$ time complexity.[4] `CarpeDiem` is never asymptotically worse than the Viterbi algorithm.

The best case happens when horizontal rewards, being equal for each transition, do not provide any discriminative power. In such a case the right hand side of the inequality in Formula 6 is zero and the inequality is guaranteed to be satisfied immediately. Moreover, being the backward strategy based on a bound similar to the one that leads to Formula 6, it will never open any other vertex. Then, a single vertex per layer is opened and `CarpeDiem` has order of $O(T + TK\log(K)) = O(TK\log(K))$ time complexity. A more formal argument about `CarpeDiem` complexity is stated by Theorem 3 and proved in Appendix B.

Theorem 3 `CarpeDiem` *has* $O(TK^2)$ *worst case time complexity and* $O(TK\log K)$ *best case time complexity.*

`CarpeDiem` finds the optimal sequence of labels. By using standard book-keeping techniques, the optimal sequence of labels can be tracked back by starting from the optimal end point. Then, the optimality of `CarpeDiem` can be proved by showing that the vertex returned by the forward strategy at the end of the algorithm is the end-point of the optimal path through the graph. This property, stated by Theorem 1, is formally proved in Appendix A.

Theorem 1 *Let us consider a sequence of calls to Algorithm 3 on layers* $2,3,\ldots,t$ $(t \leq T)$. *When Algorithm 3 terminates on layer* t*, the returned vertex* y^*_t *is the endpoint of the optimal path to layer* t*. Formally,*

$$\forall y_t : \gamma(y^*_t) \geq \gamma(y_t).$$

Beside the theoretical properties of the algorithm, it is important for the practitioner to consider its actual performances over real world problems. In the general case the algorithm will open some, but not all vertices: the exact number of the vertices that will be inspected depends on the particular

4. The $O(TK\log(K))$ term in the formula accounts for the time needed to sort vertices according to $\sqsupseteq_t$.

application and on how the features have been engineered. Empirical evidence (Section 6) suggests that many problems are closer to the best case than to the worst. Before introducing the experimentation, we show how `CarpeDiem` can be instantiated in the context of a supervised sequential learning system and, more in particular, in a system based on the voted perceptron algorithm.

5. Grounding the Voted Perceptron Algorithm on `CarpeDiem`

The supervised sequential learning problem can be formulated as follows (Dietterich, 2002).

> Let $\{(\vec{x}_i, \vec{y}_i)\}_{i=1}^{N}$ be a set of N training examples. Each example is a pair of sequences $(\vec{x}_i, \vec{y}_i)$, where $\vec{x}_i = \langle x_{i,1}, x_{i,2}, \ldots, x_{i,T_i} \rangle$ and $\vec{y}_i = \langle y_{i,1}, y_{i,2}, \ldots, y_{i,T_i} \rangle$. The goal is to construct a classifier H that can correctly predict a new label sequence $\vec{y} = H(\vec{x})$ given an input sequence $\vec{x}$.

The SSL problem has been approached with many different techniques. Among others, we recall Sliding Windows (Dietterich, 2002), hidden Markov models (Rabiner, 1989), Maximum Entropy Markov Models (McCallum et al., 2000), Conditional Random Fields (Lafferty et al., 2001), Dynamic Conditional Random Fields (Sutton et al., 2007), and the voted perceptron algorithm (Collins, 2002).

The voted perceptron uses the Viterbi algorithm at both learning and classification time. It is then particularly appropriate for the application of our technique. Moreover, it relies on the *boolean features framework* (McCallum et al., 2000) which is more general than the HMMs model with respect to representing the graph. In this framework, depending on how features are implemented, both static (homogeneous) and *dynamic* transition matrices can be modeled. We use the term *dynamic* transition matrix to indicate that weights associated to edges may change from time point to time point, depending on the observations.

In the boolean features framework the learnt classifier is built in terms of a set of boolean features. Each feature ϕ reports about a salient aspect of the sequence to be labelled in a given time instant. More formally, given a time point t, a boolean feature is a $1/0$-valued function of the whole sequence of feature vectors $\vec{x}$, and of a restricted neighborhood of y_t. The function is meant to return 1 if the characteristics of the sequence $\vec{x}$ around time step t support the classifications given at and around y_t. Under a first order Markov assumption, each ϕ depends only on y_t and y_{t-1}. Let us denote with w_ϕ the weight associated to feature ϕ. The classifier learnt by the voted perceptron algorithm has the form

$$H(\vec{x}) = \arg\max_{\vec{y}} \sum_{t=1}^{T} \sum_{\phi} w_\phi \cdot \phi(\vec{x}, y_t, y_{t-1}, t)$$

and is suitable to be evaluated using the Viterbi algorithm.

In practice, not all boolean features depend on both y_t and y_{t-1}. Let us distinguish features depending on both y_t and y_{t-1} from those depending only on y_t. We denote with Φ^0 the set of features that depends only on y_t and thus models per vertex (vertical) information. Analogously, we denote with Φ^1 the set of features that depend on both y_t and y_{t-1} modeling, thus, per edge (horizontal) information. The vertical and horizontal weights can be then calculated as:

$$S^0_{y_t} = \sum_{\phi \in \Phi^0} w_\phi \phi(\vec{x}, y_t, t)$$

and

$$S^1_{y_t,y_{t-1}} = \sum_{\phi \in \Phi^1} w_\phi \phi(\vec{x}, y_t, y_{t-1}, t).$$

In general, the bound on the maximal transition weight S^{1*} can be set to the sum of all positive horizontal weights:

$$S^{1*} = \sum_{\phi \in \Phi^1} J(w_\phi)$$

where $J(x)$ is x if $x > 0$, and 0 otherwise. It is noteworthy that this quantity can be computed without any extra—domain specific—knowledge.

Often, however, better bounds can be given based on specific domain knowledge. An example of such improvements (one that we exploit throughout our experimentation) consists in partitioning the horizontal features into sets of mutually exclusive features. Then the bound can be computed as the sum of the maximal weight of each partition. In case the partitions degenerate to a single set (i.e., all horizontal features are mutually exclusive), the maximal horizontal weight can be used. For instance, in many domains where HMMs are routinely applied, horizontal features are used only to check the last two predicted labels. In such domains, if a horizontal feature is asserted, no other feature can and we can appropriately set

$$S^{1*} = \max_{\phi \in \Phi^1} J(w_\phi). \tag{7}$$

6. Experimentation

To figure out whether and how `CarpeDiem` can be applied to actual tasks, we tested it on three different problems: the problem of music harmony analysis (Radicioni and Esposito, 2007), the frequently asked questions (FAQs) segmentation problem (McCallum et al., 2000), and a text recognition problem built starting from the "letter recognition" data set from the UCI machine learning repository (Frey and Slate, 1991).

The running time of an execution of `CarpeDiem` depends on how the weights of vertical and horizontal features compare: the more discriminative are vertical features with respect to horizontal features, the larger is the edge `CarpeDiem` has over the Viterbi algorithm.

Overall the three experiments cover three situations that are likely to occur in practice. The music analys problem represents a situation where S^{1*} has been selected by exploiting detailed domain knowledge: horizontal features have been divided into sets of non trivial partitions and the bound has been set accordingly (see end of Section 5). Features of the FAQs segmentation problem have been developed by McCallum et al. (2000) on a different system, and then imported into ours without modifications. Features used in the text recognition task didn't go through a real engineering process; on the contrary, they can be seen as a first, to some extent *naive*, attempt to tackle the problem. In these last two cases, we have set S^{1*} using Formula 7.

6.1 Tonal Harmony Analysis

Given a musical flow, the task of music harmony analysis consists in associating a label to each time point (Temperley, 2001; Pardo and Birmingham, 2002). Such labels reveal the underlying harmony by indicating a fundamental note (*root*) and a *mode*, using chord names such as 'C minor'.

Music analysis task can be naturally represented as a supervised sequential learning problem. In fact, by considering only the "vertical" aspects of musical structure, one would hardly produce reasonable analyses. Experimental evidences about human cognition reveal that in order to disambiguate unclear cases, composers and listeners refer to "horizontal" features of music as well: in these cases, context plays a fundamental role, and contextual cues can be useful to the analysis system.

The system relies on 39 features. They have been engineered so that they take into account the prescriptions from music harmony theory, a field where vertical and horizontal features naturally arise. *Vertical* features report about simultaneous sounds and their correlation with the currently predicted chord. *Horizontal* features capture metric patterns and chordal successions. This is a case where not all horizontal features are mutually exclusive (i.e., S^{1*} is not the maximal of positive horizontal weights) and where horizontal weights may change over time. For instance, the same transition between two chords can receive different weights according to whether it falls on accented/unaccented beats.

The training set is composed of 30 chorales (3,020 events) by J.S. Bach (1675-1750). The classifiers have been tested on 42 separate chorales (3,487 events) from the same author.

6.2 FAQs Segmentation

We experimented on the FAQs segmentation problem as introduced by McCallum et al. (2000). It basically consists of segmenting Usenet FAQs into four distinct sections: 'head', 'question', 'answer', and 'tail'.

In this data set, events correspond to text lines and sequences correspond to FAQs. McCallum et al. define 24 boolean features. Each one is coupled with each possible label for a total of 96 features. Additionally, 16 features are used to take into account the possible transitions between labels.

The data set consists of a learning set containing 26 sequences (29,406 events) and a test set containing 22 sequences (33,091 events).

6.3 Text Recognition

Our third experiment deals with the problem of recognizing printed text. We trained the classifiers on the "The Frog King" tale (122 sequences, 6,931 events) by Grimm brothers, and tested over the "Cinderella" tale (240 sequences, 13,354 events) by the same authors. The classifier is called to recognize each letter composing the tale. The data set has been built as follows. Each letter (corresponding to an individual event) in the tales has been encoded by picking at random one of its possible descriptions as provided by the *letters* UCI data set[5] (Frey and Slate, 1991). Each sentence corresponds to a distinct sequence.

We briefly recall here the characteristics of the letters data set as originally proposed by the authors. The data set contains 20,000 letters described using 16 integer valued features. Such attributes capture highly heterogeneous facets of the scanned raw image such as: horizontal and vertical position, the width and height, the mean number of edges per pixel row. The images have been obtained by randomly distorting 16 fonts taken from the US National Bureau of Standards. The features used by the learning system are:

5. It can be found at `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/letter-recognition`.

Experiment	Viterbi	CarpeDiem	Time Saved (%)
music Analysis	13,582	1,507	88.90%
FAQs segmentation	537	144	73.15%
letter recognition	961,969	34,244	96.44%

Table 2: CPU Time (expressed in seconds) and percentage of time saved by CarpeDiem.

1. $26 \times (16 \times 16) = 6,656$ vertical features, obtained by the original attributes devised by Frey and Slate (1991). The reported figure is explained as follows. We consider each of the 16 values of the 16 attributes in the original data set, thus resulting in 256 possible combinations. Each such combination is still to be coupled with the 26 letters of the English alphabet.

2. the $27 \times 27 = 729$ horizontal features, obtained by considering $\mathcal{A} \times \mathcal{A}$, where $\mathcal{A}$ is the set of letters in the english alphabet, plus a sign for blanks.

6.4 Procedure

We compare the performances of CarpeDiem against those provided by the Viterbi algorithm. To this aim, we embedded CarpeDiem in a SSL system implementing the voted perceptron learning algorithm (Collins, 2002). The learnt weights have been then used to build two classifiers: one based on the Viterbi algorithm, the other one based on CarpeDiem.

For each one of the three problems we divided the data into a learning set and a test set. Each learning set has been further divided into ten data sets of increasing sizes (the first one contains 10% of the data, the second one 20% of the data, and so forth). Also, each experiment has been repeated by varying the number of learning iterations from 1 to 10, for a grand total of 100 classifiers per problem. We tested each learnt classifier on the appropriate test set recording the classification time obtained by using first CarpeDiem and then Viterbi.

In the following we will indicate each one of the 100 classifiers by using two numbers separated by a colon: the former number corresponds to the size of the training set (1 standing for 10%, 2 for 20%, ..., 10 for 100%), the latter one indicates the number of iterations. For instance, the classifier 8:1 has been acquired by iterating once on 80% of the learning set.

6.5 Results

As earlier mentioned (and implied by Theorem 1), CarpeDiem performs exact inference: classifiers built on CarpeDiem provide the same answers as those built on the Viterbi algorithm.

We measured the total classification time spent by the algorithms as well as the average time *saved* by CarpeDiem with respect to the Viterbi algorithm. They are provided in Table 2: average time savings range from about 73% (FAQs segmentation) to over 96% (letter recognition). Figures 4, 5 and 6 graphically report a detailed account of each experiment. Classification times for each problem were obtained using a fixed size test set. By observing the profiles reported in the figures, it is apparent that, while the Viterbi algorithm runs in approximately constant time, CarpeDiem performances depend on the particular classifier used.

In all trials of all experiments CarpeDiem runs in a small fraction of the time needed by Viterbi.

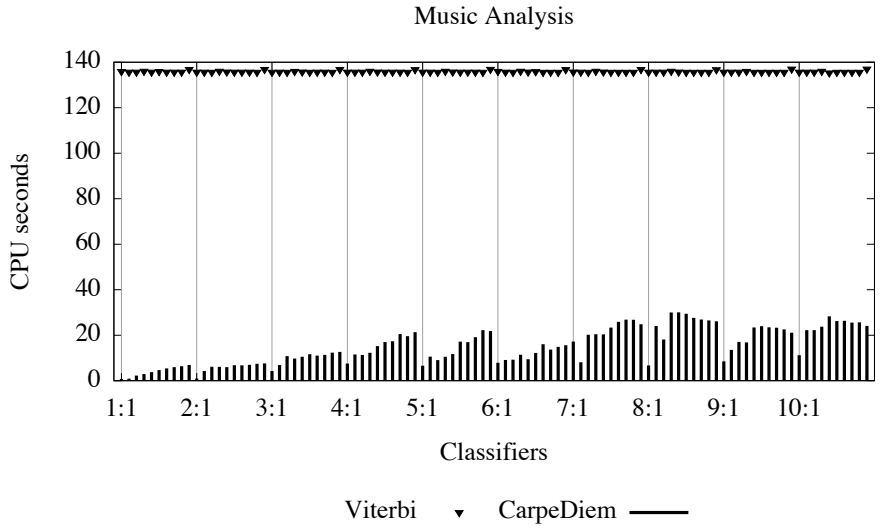


Figure 4: Results on the music analysis problem. We report on the abscissas the learnt classifiers indicating $i{:}j$ for classifier acquired using $(10 \cdot i)\%$ of the training set and j iterations. Labels on the abscissas refer only to the first classifier for each block of experiments; the following nine bars refer to the remaining ones. For instance, label 3:1 is positioned below the bar corresponding to the classifier trained on the 30% of the training set and using 1 iteration. The following nine bars refer to classifiers $3{:}2, 3{:}3, \ldots, 3{:}10$. We report on the ordinates the CPU seconds needed for the classification of the test set. Vertical bars refer to the time spent by `CarpeDiem`. Triangles refer to the time spent by Viterbi.

One interesting fact is unveiled by the profile of the classification time. Since—in each one of the three experiments—classification is performed on a data set of fixed size, one would expect roughly constant classification time. By converse, at least in the first two experiments (Figures 4 and 5), the emerging patterns are similar to those usually observed at learning time. We remark that the hundred runs of each experiment differ only in the set of weights used by the classifier. Then it is apparent that, as the voted perceptron learns, it somehow modifies the weights in a way that proves to be detrimental to the work of `CarpeDiem`.

To explain the observed patterns, let us consider an informative vertical feature $\phi_\bullet$ and examine the first iteration of the voted perceptron on a sequence of length $T = 100$. Also, we assume that $\phi_\bullet$ is asserted 60 times, and that it votes for the correct label 50 times out of 60. Even though this example may seem unrealistic, it is not.[6] The first iteration on the first sequence the voted perceptron chooses labels at random. Then, the vast majority of them will be incorrectly assigned, thus implying a large number of feature weights updates. If all the labels for which $\phi_\bullet$ is asserted are actually mislabelled, due to the way the update rule acts, the weight associated to $\phi_\bullet$ will be increased[7] by 40. This large increase occurs all at once at the end of the first iteration on the first sequence, and it is likely to overestimate the weight of $\phi_\bullet$. The voted perceptron will spend the rest of learning trying to compensate for this overestimation. However, subsequent updates will be

6. For instance, in the music analysis problem, this could be the case for the feature that votes for the chord that has exactly 3 notes asserted in the current event.

7. That is, $50 - (60 - 50) = 40$.

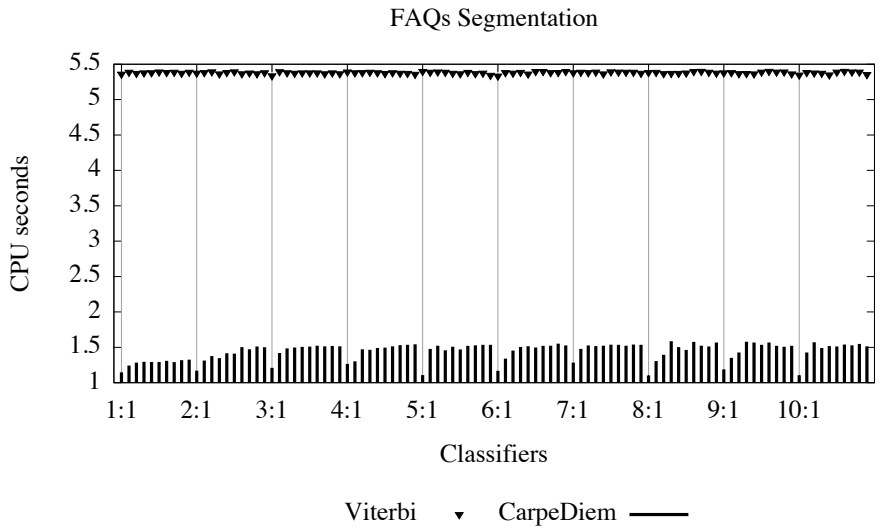


Figure 5: Results on the FAQs Segmentation problem. The conventions adopted are the same as for Figure 4.

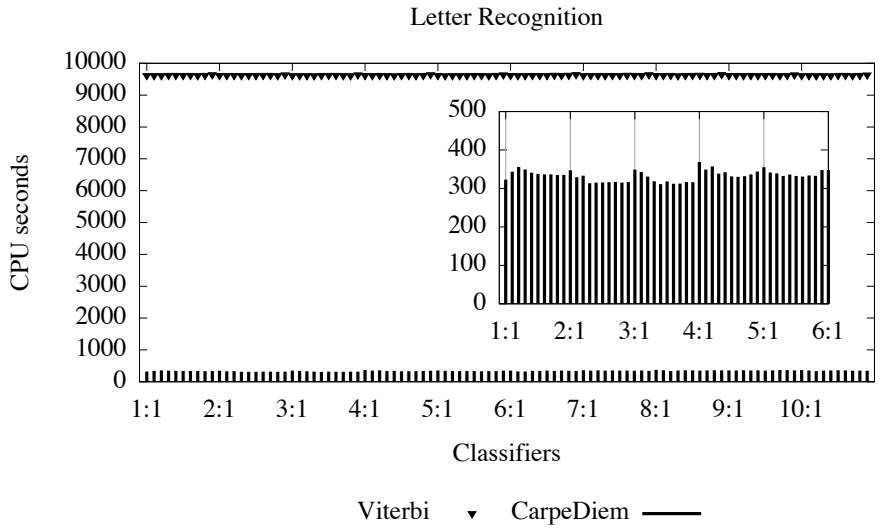


Figure 6: Results on the letter recognition problem. In the inner frame we detail the time spent by `CarpeDiem` using the first fifty classifiers. The conventions adopted are the same as for Figure 4.

of smaller magnitude. In fact, the following predicted labeling will not be randomly guessed, thus implying a reduced number of updates.

By summarizing, highly predictive features have their weights initially set to very large values; such weights slowly decrease in the following. The behavior described clearly emerges in Figure 7, where the individual weights of vertical features (for the music analysis problem) are plotted as the updates occur. Then, since `CarpeDiem` is efficient when vertical features are discriminative compared to horizontal ones, the algorithm is particularly well-suited to be used during the early

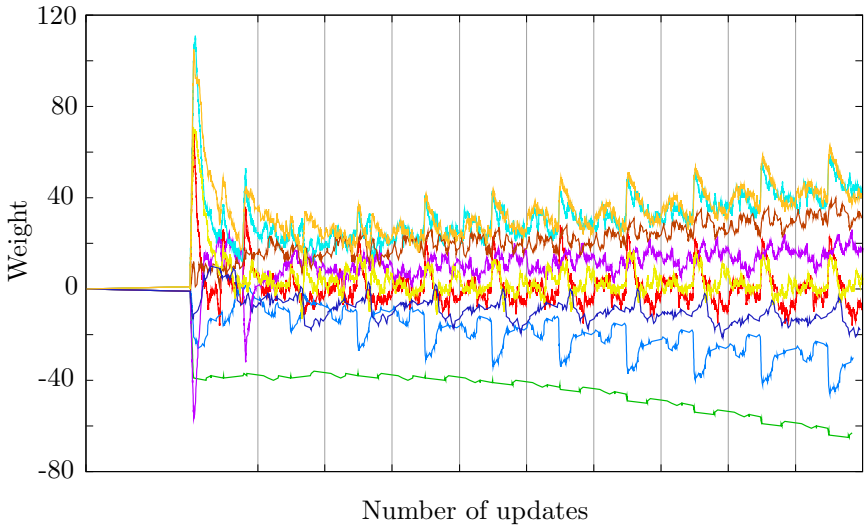


Figure 7: Evolution of vertical weights throughout learning. Each line corresponds to an individual vertical feature; vertical lines correspond to the beginning of new iterations. The plot refers to the music analysis data set.

learning steps of the voted perceptron where the time saved by `CarpeDiem` reaches peaks of 99.47% (music analysis, classifier 1:1) and 78.60% (FAQs segmentation, classifier 1:1).

Lastly, one might wonder why the observed pattern wasn't seen in the text recognition problem. Notwithstanding that the problem itself is a sequential one, to model only bigrams probabilities proved—against our expectations—to be not enough to provide sufficient discrimination power to horizontal features. In other words, the horizontal features we devised contribute to the correct classification only in a marginal way. This is a setting where `CarpeDiem` can, and actually does, attain exceptional time savings (Figure 6).

Prompted by such time savings, we re-ran the same experiments using no horizontal features: the accuracy does not drop down as significantly as in the other two problems. This fact explains why the pattern observed in Figures 4 and 5 is not observed in Figure 6: since vertical features remain predictive with respect to horizontal ones throughout the learning process the performances of `CarpeDiem` do not change over time. Here we note one important property of `CarpeDiem`: `CarpeDiem` time performances reflect the degree of sequentiality inherent to the problem at hand. Actually, by tracking the number of vertices opened in each layer, one can even measure how important sequential information is in different parts of the same sequence.

6.6 Comparison with Related Algorithms

In the following we present a brief discussion about two technologies that, if not directly comparable with `CarpeDiem`, are much related to the algorithm. We start by reporting the results of implementing the `CarpeDiem` heuristic for the A^* algorithm. Then, we report about a performance comparison with non-optimal search algorithms based on the Viterbi beam search approach.

6.6.1 RELATIONS WITH A^*

It could be argued that the `CarpeDiem` algorithm looks interestingly similar to the A^* algorithm (Hart et al., 1968). In order to investigate this similarity let us consider the following heuristic, based on the same ideas underlying `CarpeDiem`:

$$h(y_t) = \sum_{t'>t} \left[\max_{y_{t'}} \left(S^0_{y_{t'}} \right) + S^{1*} \right] = (T-t)S^{1*} + \sum_{t'>t} \left[\max_{y_{t'}} \left(S^0_{y_{t'}} \right) \right].$$

The heuristic estimates the distance to the goal by summing the best vertical weight in each layer and the best possible transition between any two layers.

Even though the heuristic is intuitively connected to the strategy implemented in `CarpeDiem`, the differences are indeed remarkable. A first important difference is in the choice criteria implemented in Algorithms 3 and 4 (we will focus only on Algorithm 3 for the sake of exposition). The criterion in Algorithm 3 states: *do not consider any further node in this layer if*

$$\mathbb{G}(y^*_t) >= \mathbb{B}_t + S^0_{y'_t}.$$

Since $\mathbb{B}_t = \mathbb{G}(y^*_{t-1}) + S^{1*}$, the above criterion approximates the reward of the optimal path by means of S^{1*} only once rather than the $T-t$ times taken by $h(y_t)$. Hence, A^* incurs an high risk of opening vertices in layers far from the goal only because of the cumulation of these approximations.

Another important difference between the two algorithms is in the fact that `CarpeDiem` implements two different heuristics: one is used by Algorithm 3 and one is used by Algorithm 4. On the contrary A^* uses the same criterion throughout the computation. Finally, the data structures needed by `CarpeDiem` are simpler (and faster) than those needed to implement A^*.

In order to empirically assess whether A^* could be competitive with respect to `CarpeDiem`, we implemented the above heuristic and paid particular attention to tuning the data structures. The resulting algorithm turned out to be even less efficient than the Viterbi algorithm.

Of course, this does not mean that the same principles implemented by `CarpeDiem` could not be plugged into A^* by means of a carefully chosen heuristic. Rather, it shows that the problem is not trivial, and deserves *ad-hoc* research efforts.

6.6.2 VITERBI BEAM SEARCH

To compare `CarpeDiem` with algorithms based on the Viterbi beam search (VBS) strategy presents several difficulties. On the one hand, we have an algorithm that guarantees optimality, but not computational performances; on the other hand, we have VBS approaches that guarantee computational performances abdicating optimality.

In VBS approaches the width of the beam (hereafter b) is particularly important in that it affects the tradeoff between computational gain and result optimality. For very small b, VBS would most probably lead to crude approximations; in this case VBS is likely to run faster than CarpeDiem just because it disregards most of the possibly-optimal paths. By converse, if we were to choose the beam size almost equal to the number of possible labels K (i.e., $b \simeq K$), then VBS would run in almost the same time as the Viterbi algorithm. In between there are all other options. Also, the quality of the solution found by VBS approaches highly depends on the heuristic adopted. In the following we will disregard any accuracy concern, so that the two approaches could be compared in terms of execution times solely.

Let ξ be the ratio between the time spent by `CarpeDiem` and the time spent by the Viterbi algorithm for solving a given problem: that is, the time savings reported in Table 2 are computed as $[(1-\xi)\times 100]\%$. Given that the time spent by a Viterbi beam search algorithm is in the order of b^2T, the time saved by a VBS algorithm w.r.t. the Viterbi algorithm is in the order of:

$$\frac{(K^2-b^2)T}{K^2T} = 1 - \frac{b^2}{K^2}\ .$$

By equating the time saved by `CarpeDiem` (i.e., $1-\xi$) and the time saved via the VBS approach, we have that by setting:

$$b = \sqrt{\xi K^2} \tag{8}$$

an algorithm based on the VBS approach will run in about the same time as `CarpeDiem`.

We implemented a basic VBS algorithm and measured its running time over the data set described in Section 6 and experimented with different settings of b. The results show that to obtain the same running times as `CarpeDiem`, the beam size needs to be set as follows:

- $b = 25$ for the Tonal Harmony Analysis data set. Being $K = 78$, this amounts to consider 32% of the possible labels;
- $b = 2$ for the FAQ Segmentation data set. Being $K = 4$, this amounts to consider 50% of the possible labels;
- $b = 5$ for the text recognition data set. Being $K = 27$, this amounts to consider 18.5% of the possible labels;

Two aspects are remarkable in the above results: *i)* the reported figures have been observed empirically rather than determined using the above formula. It is immediate to verify that they are very close to the ones returned by using Formula 8; *ii)* we omitted to implement a heuristic to guide the VBS search. Since computing the heuristic would require additional efforts, the timings used to derive such numbers are optimistic approximations of the actual time needed by a full fledged VBS algorithm.

In summary, `CarpeDiem` runs at least as fast as a VBS approach when a small to medium beam size is used while providing some additional benefits. Again, investigating the conditions that make appropriate one approach over the other one, deserves further deepening the research.

7. Conclusions

In this paper we have proposed `CarpeDiem`, a replacement of the Viterbi algorithm. On average, `CarpeDiem` allows evaluating the best path in a layered and weighted graph in a fraction of the time needed by the Viterbi algorithm. `CarpeDiem` is based on a property exhibited by most tasks commonly tackled by means of sequential learning techniques: the observations at and around a time instant t are very relevant for determining the t-th label, while information about the succession of labels is mainly useful in order to disambiguate unclear cases. The extent to which the property holds determines the performances of the algorithm. We formally proved that `CarpeDiem` finds the best possible sequence of labels *and* that the algorithm complexity ranges between $O(TK\log K)$ in the best case, and $O(TK^2)$ in the worst case. The fact that the worst case complexity is the same as the Viterbi algorithm complexity suggests that, by and large, `CarpeDiem` is suitable for substituting the Viterbi algorithm.

In Section 3, we reviewed recently proposed alternatives and pointed out the following advantages of CarpeDiem with respect to the competitors:

- it is parameter free;
- it does not require any prior knowledge or tuning;
- it never compromises on optimality.

At the present time, the main strength of other approaches with respect to ours is that they have been devised to improve the forward-backward algorithm as well. We defer the extension of our approach in that direction to future work.

In addition to the theoretical grounding of CarpeDiem complexity, we provided an experimentation on three real world problems, and compared its execution time with that of the Viterbi algorithm. The experiments show that large time savings can be obtained. The reported figures show time savings ranging from 73% to 96% with respect to the Viterbi algorithm.

A further interesting facet of CarpeDiem is that its execution trace provides clues about the problems at hand. It can then be used to understand how salient sequential information is. Also, within the same sequence, it is often interesting to investigate the time steps where sequential information gets crucial thus implying higher classification time. For instance, in Radicioni and Esposito (2007) we used this fact to find where musical excerpts get more difficult.

In a world where the quantity of information as well as its complexity is ever increasing, there is the need for sophisticated tools to analyse it in reasonable time. Our perception is that, in most problems, long chains of dependencies are useful to reach top results, but their influence on the correct labelling decreases with the length of the chain. In a probabilistic setting, the length of the chain of dependencies would be called the 'order of the Markov assumption'. In such a context, it would be appropriate to say that what we call 'vertical' features are actually features making a zero order Markov assumption (no dependency at all), while horizontal features are features making a first order Markov assumption. The higher the order of the Markov assumption we want to plug into the model, the slower the algorithm that evaluates the sequential classifier. If our guess is correct, however, the higher the Markov assumption, the less informative are the features that use it. Should this be the case, CarpeDiem strategy could be extended to efficiently handle higher order Markov assumptions, thereby allowing to use sequential classifiers to tackle a larger set of problems.

Acknowledgments

We wish to thank Luca Anselma, Guido Boella, Leonardo Lesmo, and Lorenza Saitta for their precious advices on preliminary versions of the work. We also want to thank the anonymous reviewers for their helpful comments and suggestions.

Appendix A. Soundness

A proof of soundness for CarpeDiem consists in showing that y_T^* is the endpoint of the optimal path through the graph of interest. Although the whole algorithm is concerned with finding out the optimal path, we presently restrict ourselves to find the optimal endpoint, since standard book

symbol	description
$S^0_{y_t}$	vertical weight of vertex y_t
$S^1_{y_t,y_{t-1}}$	horizontal weight for transition from y_{t-1} and y_t
S^{1*}	maximal transition reward (fixed for the whole graph)
$\gamma(y_t)$	the weight of the best path to y_t
γ^*_t	the weight of the best path to the best vertex in level t
β_t	$\gamma^*_{t-1} + S^{1*}$
$\mathbb{G}(y_t)$	$\gamma(y_t)$ as calculated by `CarpeDiem`
$\mathbb{B}_t$	β_t as calculated by `CarpeDiem`

Table 3: Summary of the notation adopted.

keeping techniques can be used during the search to store path information. The path can be then be retrieved in $O(T)$ time.

The proof consists in two Theorems and one Lemma; in particular, Theorem 1 directly implies the soundness of `CarpeDiem`, while Theorem 2 and Lemma 1 are necessary in the proof of Theorem 1.

Before entering the core of the proof, let us summarize the notation adopted. We distinguish between values that are calculated by `CarpeDiem`, and those representing properties of the graph. We will use blackboard characters ($\mathbb{G}$ and $\mathbb{B}$) to denote the former ones and greek letters (γ and β) for the latter ones. A summary of important definitions is reported in Table 3.

We start by stating and proving Lemma 1, which ensures the soundness of the main bound used by `CarpeDiem`.

Lemma 1 *If $\mathbb{B}_t = \beta_t$ then the bound exploited by CarpeDiem does not underestimate the reward of the optimal path to any vertex. Formally,*

$$\mathbb{B}_t = \beta_t \Rightarrow \mathbb{B}_t + S^0_{y_t} \geq \gamma(y_t).$$

Proof Let us consider the optimal path to y_t and denote with $\pi(y_t)$ the predecessor of y_t. Then, by definition we have: $\gamma^*_{t-1} \geq \gamma(\pi(y_t))$, and $S^{1*} \geq S^1_{y_t,\pi(y_t)}$. It immediately follows that:

$$\gamma^*_{t-1} + S^{1*} + S^0_{y_t} \geq \gamma(\pi(y_t)) + S^1_{y_t,\pi(y_t)} + S^0_{y_t}.$$

By definition $\beta_t = \gamma^*_{t-1} + S^{1*}$ and $\gamma(y_t) = \gamma(\pi(y_t)) + S^1_{y_t,\pi(y_t)} + S^0_{y_t}$, which yields:

$$\beta_t + S^0_{y_t} \geq \gamma(y_t)$$

and by assumption, this implies:

$$\mathbb{B}_t + S^0_{y_t} \geq \gamma(y_t).$$

■

Theorem 1 *Let us consider a sequence of calls to Algorithm 3 on layers $2,3,\ldots,t$ $(t \leq T)$. When Algorithm 3 terminates on layer t, the returned vertex y^*_t is the endpoint of the optimal path to layer t. Formally,*

$$\forall y_t : \gamma(y^*_t) \geq \gamma(y_t).$$

Proof We prove the stronger fact

$$\mathbb{B}_t = \beta_t \wedge \forall y_t : \gamma(y_t^*) \geq \gamma(y_t).$$

The proof is by induction on t. The base case for the induction is guaranteed by the initialization step in Algorithm 2 where $\mathbb{B}_2$ and $\gamma(y_1^*)$ are set. We start by showing that

$$\forall y_1 : \gamma(y_1^*) \geq \gamma(y_1) \tag{9}$$

as follows:

$$\begin{array}{rcl}
\gamma(y_1^*) = & \gamma\left(\arg\max_{y_1} \mathbb{G}(y_1)\right) & \rightarrow \text{by line 3 (Algorithm 2)} \\
= & \gamma\left(\arg\max_{y_1} S^0_{y_1}\right) & \rightarrow \text{by line 2 (Algorithm 2)} \\
= & \gamma\left(\arg\max_{y_1} \gamma(y_1)\right) & \rightarrow \text{by Equation 2} \\
= & \max_{y_1} \gamma(y_1) & \\
& \Downarrow & \\
& \forall y_1 : \gamma(y_1^*) \geq \gamma(y_1). &
\end{array}$$

In order to prove $\mathbb{B}_2 = \beta_2$, we note that Algorithm 2 sets $\mathbb{B}_2$ to $\mathbb{G}(y_1^*) + S^{1*}$:

$$\begin{array}{rcl}
\mathbb{B}_2 = & \mathbb{G}(y_1^*) + S^{1*} & \\
= & S^0_{y_1^*} + S^{1*} & \rightarrow \text{by line 2 (Algorithm 2)} \\
= & \gamma(y_1^*) + S^{1*} & \rightarrow \text{by Equation 2} \\
= & \gamma_1^* + S^{1*} & \rightarrow \text{by definition of } \gamma_1^* \text{ (Table 3) and Equation 9} \\
= & \beta_2 & \rightarrow \text{by definition of } \beta_2 \text{ (Table 3).}
\end{array}$$

Let us now assume that for all $\hat{t}$, $1 \leq \hat{t} < t$:[8]

$$\begin{array}{c}
\mathbb{B}_{\hat{t}} = \beta_{\hat{t}} \\
\forall y_{\hat{t}} : \gamma(y_{\hat{t}}^*) \geq \gamma(y_{\hat{t}})
\end{array}$$

then we prove $\mathbb{B}_t = \beta_t$ as follows:

$$\begin{array}{rcl}
\mathbb{B}_t = & \mathbb{G}(y_{t-1}^*) + S^{1*} & \rightarrow \text{by instruction 6 (Algorithm 3)} \\
= & \gamma(y_{t-1}^*) + S^{1*} & \rightarrow \text{by Theorem 2} \\
= & \gamma_{t-1}^* + S^{1*} & \rightarrow \text{by Equation 10 and definition of } \gamma_{t-1}^* \\
= & \beta_t & \rightarrow \text{by definition of } \beta_t \text{ (Table 3).}
\end{array}$$

In order to prove $\forall y_t : \gamma(y_t^*) \geq \gamma(y_t)$ we start by noting that at the end of the main loop of Algorithm 3 it holds $\mathbb{G}(y_t^*) \geq \mathbb{B}_t + S^0_{y_t'}$. Also, for any $y_t \sqsubseteq_t y_t'$ we have (by Definition 2–Equation 5): $\mathbb{B}_t + S^0_{y_t} \leq \mathbb{B}_t + S^0_{y_t'}$. It follows that $y_t \sqsubseteq_t y_t' \Rightarrow \mathbb{G}(y_t^*) \geq \mathbb{B}_t + S^0_{y_t}$. Using Lemma 1 we have

$$y_t \sqsubseteq_t y_t' \Rightarrow \mathbb{G}(y_t^*) \geq \gamma(y_t). \tag{10}$$

8. We note that our definitions give no meaning to $\mathbb{B}_1$ and β_1. We define them to be equal regardless their value: this simplifies the discussion allowing an easier formulation of the properties being stated and proved. They are not used in the algorithm nor in the argument anyway; the definition is thus safe.

Moreover, since the algorithm scans the vertices in the order given by $\sqsupseteq_t$, all vertices y_t, $y_t \sqsupseteq_t y'_t$ have been considered by the main loop. Then by line 4 (Algorithm 3), by our inductive hypothesis ($\forall \hat{t} < t : \mathbb{B}_{\hat{t}} = \beta_{\hat{t}}$) and Theorem 2, we have that for each such vertex $\mathbb{G}(y_t) = \gamma(y_t)$. Moreover, due to line 5 (Algorithm 3), $\mathbb{G}(y_t^*) \geq \mathbb{G}(y_t)$. Putting together the two statements, we conclude that

$$y_t \sqsupseteq_t y'_t \Rightarrow \mathbb{G}(y_t^*) \geq \gamma(y_t). \tag{11}$$

Equation 10, Equation 11, and the fact that $\sqsupseteq_t$ is a total order, yield

$$\forall y_t : \mathbb{G}(y_t^*) \geq \gamma(y_t).$$

By noting that y_t^* is open (and exploiting again Theorem 2), we have:

$$\forall y_t : \gamma(y_t^*) \geq \gamma(y_t).$$

■

Theorem 2 *Let us assume $\forall \hat{t} < t : \mathbb{B}_{\hat{t}} = \beta_{\hat{t}}$, then after opening vertex y_t, $\mathbb{G}(y_t) = \gamma(y_t)$.*

Proof By line 7 (Algorithm 4), $\mathbb{G}(y_t) = \mathbb{G}(y_{t-1}^*) + S^1_{y_t, y_{t-1}^*} + S^0_{y_t}$. Then, our main goal is to prove

$$\mathbb{G}(y_{t-1}^*) + S^1_{y_t, y_{t-1}^*} + S^0_{y_t} = \gamma(y_t).$$

Replacing $\gamma(y_t)$ with its definition (Equation 2) yields:

$$\mathbb{G}(y_{t-1}^*) + S^1_{y_t, y_{t-1}^*} + S^0_{y_t} = \max_{y_{t-1}} \left(\gamma(y_{t-1}) + S^1_{y_t, y_{t-1}} + S^0_{y_t} \right).$$

The above equality is satisfied if the following two properties hold:

$$y_{t-1}^* = \arg\max_{y_{t-1}} \left(\gamma(y_{t-1}) + S^1_{y_t, y_{t-1}} \right) \tag{12}$$

$$\mathbb{G}(y_{t-1}^*) = \gamma(y_{t-1}^*). \tag{13}$$

The proof, by induction on t, proves that the Equations 12 and 13 are satisfied at the moment (and after) $\mathbb{G}(y_t)$ is set. `CarpeDiem` starts by opening the most promising vertex in layer 2, this is the first time Algorithm 4 is called and hence the base case of the induction. Let us consider what happens when a node y_2 is opened. Since all vertices in layer 1 have been opened by the initialization step, the first loop in Algorithm 4 iterates on all of them and the second loop is never entered. Then, just before line 7, it holds

$$y_1^* = \arg\max_{y_1} \left(\mathbb{G}(y_1) + S^1_{y_2, y_1} \right).$$

Since the initialization step guarantees $\forall y_1 : \mathbb{G}(y_1) = \gamma(y_1)$, then properties 12 and 13 are satisfied.

Let us now assume by induction that after opening a vertex y_{t-1} in layer $t-1$ ($t > 2$) it holds $\mathbb{G}(y_{t-1}) = \gamma(y_{t-1})$. We focus on the execution of Algorithm 4 on a vertex y_t in layer t. Let us denote with O_{t-1} the set of vertices presently open in layer $t-1$, and with C_{t-1} the set of closed ones. When the first loop ends, it holds:

$$y_{t-1}^* = \arg\max_{y_{t-1} \in \mathsf{O}_{t-1}} \left(\mathbb{G}(y_{t-1}) + S^1_{y_t, y_{t-1}} \right).$$

Also, since all vertices for which we have taken the arg max are in layer $t-1$ and open, we apply the inductive hypothesis and conclude that:

$$y^*_{t-1} = \arg \max_{y_{t-1} \in \mathsf{O}_{t-1}} \left(\gamma(y_{t-1}) + S^1_{y_t, y_{t-1}} \right). \tag{14}$$

The second loop moves some vertices from C_{t-1} to O_{t-1}. At the same time, however, it updates y^*_{t-1} so that the above equality is preserved. Then, on exit we can conclude (14) and (for the particular y'_{t-1} that caused the loop to exit):

$$\mathbb{G}(y^*_{t-1}) + S^1_{y_t, y^*_{t-1}} \geq \mathbb{B}_{t-1} + S^0_{y'_{t-1}} + S^{1*}. \tag{15}$$

Also, by definition of $\sqsupseteq_{t-1}$ (Definition 2–Equation 5), $\forall y_{t-1} : y'_{t-1} \sqsupseteq_{t-1} y_{t-1}$ implies:

$$\mathbb{B}_{t-1} + S^0_{y'_{t-1}} + S^{1*} \geq \mathbb{B}_{t-1} + S^0_{y_{t-1}} + S^{1*}. \tag{16}$$

Since vertices are considered in $\sqsupseteq_{t-1}$ order and since y'_{t-1} is the first vertex that has not been opened, it follows that all closed vertices follow y'_{t-1} in the $\sqsupseteq_{t-1}$ order. Using this fact along with (15) and (16), it follows:

$$\forall y_{t-1} \in \mathsf{C}_{t-1} : \mathbb{G}(y^*_{t-1}) + S^1_{y_t, y^*_{t-1}} \geq \mathbb{B}_{t-1} + S^0_{y_{t-1}} + S^{1*}.$$

By induction, $\mathbb{G}(y^*_{t-1}) = \gamma(y^*_{t-1})$. Moreover, Lemma 1 implies $\mathbb{B}_{t-1} + S^0_{y_{t-1}} \geq \gamma(y_{t-1})$. Using these facts, along with $\forall y_t, y_{t-1} : S^{1*} \geq S^1_{y_t, y_{t-1}}$ (Definition 2–Equation 3) we obtain:

$$\forall y_{t-1} \in \mathsf{C}_{t-1} : \gamma(y^*_{t-1}) + S^1_{y_t, y^*_{t-1}} \geq \gamma(y_{t-1}) + S^1_{y_t, y_{t-1}}$$

Which yields:

$$\gamma(y^*_{t-1}) + S^1_{y_t, y^*_{t-1}} \geq \max_{y_{t-1} \in \mathsf{C}_{t-1}} \left(\gamma(y_{t-1}) + S^1_{y_t, y_{t-1}} \right).$$

This and (14) yield:

$$y^*_{t-1} = \arg \max_{y_{t-1}} \left(\gamma(y_{t-1}) + S^1_{y_t, y_{t-1}} \right).$$

Also, the fact that y^*_{t-1} is open and the inductive hypothesis, yield $\mathbb{G}(y^*_{t-1}) = \gamma(y^*_{t-1})$.

■

Appendix B. Complexity

Theorem 3 `CarpeDiem` *has $O(TK^2)$ worst case time complexity and $O(TK \log K)$ best case time complexity.*

Proof Let us consider the final step of an execution of `CarpeDiem`, and assume that for each layer t, exactly k_t vertices have been opened. In our proof we separately consider the time spent to process each layer of the graph. We define the quantity $\mathcal{T}(t)$ to represent the overall time spent by Algorithms 3 and 4 to process layer t. Let us define:

$a(y_t)$**:** the number of steps needed by Algorithm 3 to process vertex y_t;

$b(y_t)$: the number of steps needed by Algorithm 4 to find the best parent for node y_t.

We note that $a(y_t)$ does not include the time spent by Algorithm 4 since such time is accounted for by $b(y_t)$. Similarly, $b(y_t)$ does not include neither the time spent by Algorithm 3, nor the time spent by recursive calls to Algorithm 4. In fact, the time spent in recursive calls is taken into account by b values of vertices in previous layers. Then we can compute $\mathcal{T}(t)$ as:

$$\mathcal{T}(t) = \sum_{y_t} a(y_t) + b(y_t)$$

The total complexity of `CarpeDiem` is then:

$$\mathcal{T}(\texttt{CarpeDiem}) = \text{time for initialization} + \sum_{t=2}^{T} (O(1) + \mathcal{T}(t))$$

where the "time for initialization" includes the $O(K)$ time spent in the first loop of Algorithm 2 plus the $O(TK \log K)$ time needed to sort each layer according to $\sqsupseteq_t$. It follows:

$$\mathcal{T}(\texttt{CarpeDiem}) = O(TK \log K) + \sum_{t=2}^{T} (O(1) + \mathcal{T}(t)). \tag{17}$$

Let us now note that $a(y_t)$ is at worst $O(k_t)$. In fact, since only k_t vertices have been opened at the end of the algorithm, it follows that the steps needed to analyse a vertex y_t by (the loop in) Algorithm 3 is at most k_t. We notice that we are overestimating the cost to analyze each node since k_t is the *overall* number of iterations performed by the mentioned loop. However this overestimation simplifies the following argument without hindering the result.

$b(y_t)$ is, at worst, $O(k_{t-1})$. In fact, since only k_{t-1} vertices have been opened at the end of the algorithm, it follows that the two loops in Algorithm 4 iterate altogether at most k_{t-1} times. Moreover, since the steps performed by recursive calls are not to be included in $b(y_t)$, it follows that all operations are $O(1)$, and the complexity accounted for by $b(y_t)$ is $O(k_{t-1})$. In both cases no computational effort is spent to process closed nodes.

From the above discussion it follows that:

$$\begin{aligned}
\mathcal{T}(t) &= \sum_{y_t} a(y_t) + b(y_t) \\
&= \sum_{y_t \text{ in open vertices}} O(k_t) + O(k_{t-1}) \\
&= k_t \cdot (O(k_t) + O(k_{t-1})) \\
&= O(k_t^2 + k_t k_{t-1}).
\end{aligned}$$

Putting together the above equation and Equation 17 we have:

$$\begin{aligned}
\mathcal{T}(\texttt{CarpeDiem}) &= O(TK \log K) + \sum_{t=2}^{T} \left(O(1) + O(k_t^2 + k_t k_{t-1})\right) \\
&= O(TK \log K) + \sum_{t=2}^{T} O(k_t^2 + k_t k_{t-1}).
\end{aligned}$$

The worst case occurs when `CarpeDiem` opens every node in every layer. In such case: $k_t = K$ for each t and the above formula reduces to $O(TK^2)$. In the best case `CarpeDiem` opens only one node per layer, $k_t = 1$ for each t and the complexity is $O(TK \log K)$. ■

References

Steve Austin, Pat Peterson, Paul Placeway, Richard Schwartz, and Jeff Vandergrift. Toward a real-time spoken language system using commercial hardware. In *HLT '90: Proceedings of the Workshop on Speech and Natural Language*, pages 72–77, Hidden Valley, Pennsylvania, 1990.

John S. Bridle, Michael D. Brown, and Richard M. Chamberlain. An algorithm for connected word recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP 1982*, pages 899–902, San Francisco, CA, USA, 1982.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–8, Philadelphia, PA, 2002. Association for Computational Linguistics.

Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 111–118, Barcelona, Spain, 2004. Association for Computational Linguistics.

Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Massachussets, 1990.

Thomas G. Dietterich. Machine learning for sequential data: A review. In T. Caelli, editor, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 15–30, Windsor, Ontario, Canada, 2002. Springer-Verlag.

Thomas G. Dietterich, Pedro Domingos, Lise Getoor, Stephen Muggleton, and Prasad Tadepalli. Structured machine learning: The next ten years. *Machine Learning*, 73(1):3–23, 2008.

Robert M. Fano. A heuristic discussion of probabilistic decoding. *IEEE Transactions on Information Theory*, 9:64–73, 1963.

Pedro F. Felzenszwalb, Daniel P. Huttenlocher, and Jon M. Kleinberg. Fast algorithms for large-state-space HMMs with applications to web usage analysis. In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

Peter W. Frey and David J. Slate. Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, 6(2):161–182, 1991.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of Systems Science and Cybernetics*, 4:100–107, 1968.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

Bruce Lowerre and Raj Reddy. *Trends in Speech Recognition*, chapter The Harpy Speech Understanding System, pages 340–360. Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 591–598, Stanford, California, USA, 2000. Morgan Kaufmann.

Shay Mozes, Oren Weimann, and Michal Ziv-Ukelson. Speeding up HMM decoding and training by exploiting sequence repetitions. *Combinatorial Pattern Matching*, 4580:4–15, 2007.

Hermann Ney, Dieter Mergel, Andreas Noll, and Annedore Paeseler. Data driven search organization for continuous speech recognition. *IEEE Transactions on Signal Processing*, 40:272–281, 1987.

Hermann Ney, Reinhold Haeb-Umbach, Bach-Hiep Tran, and Martin Oerder. Improvements in beam search for 10000-word continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 9–12, San Francisco, California, USA, 1992.

Bryan Pardo and William P. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26:27–49, 2002.

Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 267–296, San Francisco, CA, USA, February 1989. Morgan Kaufmann. ISBN 1-55860-124-4.

Daniele P. Radicioni and Roberto Esposito. Tonal harmony analysis: a supervised sequential learning approach. In M. T. Pazienza and R. Basili, editors, *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing, 10th Congress of the Italian Association for Artificial Intelligence*. Springer-Verlag, 2007.

Sajid M. Siddiqi and Andrew W. Moore. Fast inference and learning in large-state-space HMMs. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 800–807, Bonn, Germany, 2005. ACM.

James C. Spohrer, Peter F. Brown, Peter H. Hochschild, and James K. Baker. Partial traceback in continuous speech recognition. In *Proceedings of the IEEE Interntational Conference on Cybernetics and Society*, pages 36–42, Boston, Massachussets, USA, 1980.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8:693–723, 2007.

David Temperley. *The Cognition of Basic Musical Structures*. MIT, Cambridge, MASS, 2001.

Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In *IEEE Transaction on Information Theory*, volume 13, pages 260–269, 1967.

Yuehua Xu and Alan Fern. On learning linear ranking functions for beam search. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning*, pages 1047–1054, Corvallis, Oregon, USA, 2007. ACM.

Journal of Machine Learning Research 10 (2009) 1881-1911 Submitted 1/09; Revised 6/09; Published 8/09

Learning Acyclic Probabilistic Circuits Using Test Paths

Dana Angluin DANA.ANGLUIN@YALE.EDU
James Aspnes JAMES.ASPNES@YALE.EDU
Department of Computer Science
Yale University
New Haven, CT 06520

Jiang Chen CRIVER@GMAIL.COM
Yahoo! Inc.
701 First Avenue
Sunnyvale, CA 94086

David Eisenstat EISENSTATDAVID@GMAIL.COM
55 Autumn St.
New Haven, CT 06511

Lev Reyzin LEV.REYZIN@YALE.EDU
Department of Computer Science
Yale University
New Haven, CT 06520

Editor: Rocco Servedio

Abstract

We define a model of learning probabilistic acyclic circuits using value injection queries, in which fixed values are assigned to an arbitrary subset of the wires and the value on the single output wire is observed. We adapt the approach of using test paths from the Circuit Builder algorithm (Angluin et al., 2009) to show that there is a polynomial time algorithm that uses value injection queries to learn acyclic Boolean probabilistic circuits of constant fan-in and log depth. We establish upper and lower bounds on the attenuation factor for general and transitively reduced Boolean probabilistic circuits of test paths versus general experiments. We give computational evidence that a polynomial time learning algorithm using general value injection experiments may not do much better than one using test paths. For probabilistic circuits with alphabets of size three or greater, we show that the test path lemmas (Angluin et al., 2009, 2008b) fail utterly. To overcome this obstacle, we introduce function injection queries, in which the values on a wire may be mapped to other values rather than just to themselves or constants, and prove a generalized test path lemma for this case.

Keywords: nonadaptive learning algorithms, probabilistic circuits, causal Bayesian networks, value injection queries, test paths

1. Introduction

Probabilistic networks are used as models in a variety of domains, for example, gene interaction networks, social networks and causal reasoning. In a binary model of gene interaction, the state of each gene is either active or inactive, and the state of each gene is determined as a function of the states of some number of other genes, its inputs. In a probabilistic variant of the model, the activation function specifies, for each possible combination of the states of the inputs, the probability that the

gene will be active (Friedman et al., 2000). In the independent cascade model of social networks, the state of each agent is active or inactive and for each pair (u,v) of agents, there is a probability that the activation of u will cause v to become active. Kempe, Kleinberg, and Tardos (2003, 2005) study the problem of maximizing influence in this and related models of social networks. In a Bayesian network there is an acyclic directed graph and a joint probability distribution over the node values such that the joint distribution is the product of each of the marginal distributions for each node given the values of the parents (in-neighbors) of the node.

A fundamental question is how much we can infer about the properties and structure of such networks from observing and experimenting with their behaviors. Prior research gives evidence from cryptography that there may be no polynomial time algorithm to learn Boolean functions represented by acyclic circuits of constant fan-in and depth $O(\log n)$ when we can set only the inputs of the circuit and observe only the output (Angluin and Kharitonov, 1995). In this paper we consider a different setting, **value injection queries**, in which we can fix the values on any subset of wires in the target circuit, but still only observe the output of the circuit.

The concept of value injection queries was inspired by models of gene suppression and gene overexpression in the study of gene interaction networks (Akutsu et al., 2003; Ideker et al., 2000) and was introduced by Angluin et al. (2009). In a causal Bayesian network there is an additional action $\mathrm{do}(X = x)$ that forces a node X to take on a value x (Pearl, 2000). A value injection query may also be viewed as a set of such actions, one for each wire fixed to a value.

Angluin et al. (2009) investigate the learnability of deterministic circuits using value injection queries and behavioral equivalence queries. Polynomial time learning algorithms using just value injection queries are given for two classes of acyclic circuits. Circuit Builder uses value injection queries to learn acyclic deterministic circuits with constant-size alphabets, constant fan-in and depth $O(\log n)$ up to behavioral equivalence in polynomial time. Another algorithm is given that learns constant-depth acyclic Boolean circuits with NOT gates and unbounded fan-in AND, OR, NAND and NOR gates up to behavioral equivalence in polynomial time using value injection queries. Negative results include an exponential lower bound on the number of value injection queries to learn acyclic Boolean circuits of unbounded depth and unbounded fan-in, and the **NP**-hardness of learning acyclic Boolean circuits of unbounded depth and constant fan-in using value injection queries.

In extending these results to analog circuits, Angluin et al. (2008b) consider circuits with polynomial-size alphabets. They give evidence of the computational hardness of learning acyclic circuits over a polynomial-size alphabet even if the depth is restricted to $O(\log n)$, motivating structural restrictions on the graphs of the circuits to achieve polynomial time learnability. They give the Distinguishing Paths Algorithm, which uses value injection queries and learns acyclic deterministic circuits that are transitively reduced and have polynomial-size alphabets, constant fan-in and unbounded depth up to behavioral equivalence in polynomial time. They also give a generalization to circuits with a constant bound on shortcut width.

In this paper we seek to extend some of these positive learnability results to the case of acyclic probabilistic circuits. The key technique in the previous work has been the idea of a **test path** for an arbitrary wire w in the circuit. Informally speaking, a test path is a directed path of wires from w to the output wire in which each wire is an input of the next wire on the path, and the other (non-path) inputs of wires on the path are fixed to constant values, thus isolating the wires along the path from the rest of the circuit. Ideally, the choice of constant values is made in such a way as to maximize the effect on the output of the circuit of changing w from one value to another. A test path thus functions as a kind of "microscope" for viewing the effects of assigning different values to the

wire w. The primary focus of this paper is to understand the properties of test paths in probabilistic circuits, and the extent to which they can be used to give polynomial time algorithms for learning probabilistic acyclic circuits.

In Section 2 we formally define our model of acyclic probabilistic circuits, value injection queries and distribution injection queries, behavioral equivalence, and the learning problem that we consider. In Section 3 we establish some basic results about probabilistic circuits and value and distribution injection experiments. In Section 4 we review the test path lemma used in previous work to establish the ability of a learner to infer circuit behavior from a small subset of experiments and show that it fails utterly in probabilistic circuits with alphabet size greater than two. However, for Boolean probabilistic circuits, we show that the test path lemma holds with an attenuation factor that depends on the structure of the circuit. (Lemma 10 treats general acyclic circuits and Corollary 11 specializes the bound to transitively reduced circuits.) In Section 5 we apply the test path lemma in the Boolean case to adapt the Circuit Builder algorithm (Angluin et al., 2009) to find using value injection queries, with high probability, in time polynomial in n and $1/\varepsilon$, a circuit that is ε-behaviorally equivalent to a target acyclic Boolean probabilistic circuit of size n with constant fan-in and depth bounded by a constant times $\log n$. In Section 6, we consider lower bounds on the attenuation of paths; Theorem 16 shows that our bound is tight for transitively reduced circuits and Theorem 18 gives a lower bound for the case of general acyclic circuits. In Section 7 we give evidence that polynomial time algorithms using general value injection experiments may not do significantly better than algorithms that use test paths. In Section 8 we introduce a stronger kind of query, a **function injection query**, and show that test paths with function injections overcome the limitations of test paths for circuits with alphabets of size greater than two.

2. Model

We extend the circuit learning model (Angluin et al., 2008b, 2009) to probabilistic gates. An unusual feature of this model is that circuits do not have distinguished inputs—since the learning algorithm seeks to predict the output behavior of value injection experiments that override the values on an arbitrary subset of wires, each wire is a potential input.

2.1 Probabilistic Circuits

A **probabilistic circuit** C of **size** $n \geq 1$ has n **wires**, of which one is the distinguished **output wire**. We call the set of C's wires W, and these wires take values in a finite **alphabet** Σ with $|\Sigma| \geq 2$. If $\Sigma = \{0,1\}$, then C is **Boolean**. The value on a wire is ordinarily determined by the output of an associated probabilistic gate, whose distribution is a function of the values on other wires.

Formally, a **value distribution** D is a probability distribution over Σ, that is, a map from Σ to the real interval $[0,1]$ such that $\sum_{\sigma\in\Sigma} D(\sigma) = 1$. The probability of σ is $D(\sigma)$. The **support** of D is the set of values $\sigma \in \Sigma$ such that $D(\sigma) > 0$. When the support of D is a singleton $\{\sigma\}$, we say D is **deterministic**. For a nonempty set of values $S \subseteq \Sigma$, the **uniform distribution** $U(S)$ is the distribution such that $U(S)(\sigma) = [\sigma \in S]/|S|$, that is, has value 0 on $\sigma \notin S$ and $1/|S|$ for $\sigma \in S$.

A k-ary **probabilistic gate function** f maps each k-tuple of values $(\sigma_1, \ldots, \sigma_k) \in \Sigma^k$ to a value distribution. When C is Boolean, we can specify f by a truth table giving the expected value for each Boolean vector of inputs. A probabilistic gate function is **deterministic** if it maps k-tuples to deterministic value distributions only.

A **probabilistic gate** g of **fan-in** k pairs a k-ary probabilistic gate function f with a k-tuple $(w_1,\ldots,w_k) \in W^k$ of **input wires**. The gate g is **deterministic** if its gate function f is deterministic. When $k = 0$, the gate g has no inputs, and we can regard it as specifying a value distribution, or, when C is Boolean, a biased coin flip.

A **probabilistic circuit** C maps wires to probabilistic gates. C is **deterministic** if all of its gates are deterministic. The **fan-in** of C is the maximum fan-in over C's gates. The **circuit graph** of C has a node for each wire in W and a directed edge (u,w) if u is one of the input wires of the gate associated with w. It is important to distinguish between wires in the circuit and edges in the circuit graph. For example, if wire u is an input of wires v and w, then there will be two directed edges, (u,v) and (u,w), in the circuit graph.

Wire w is **reachable** from wire u if there is a directed path from u to w in the circuit graph. A wire is **relevant** if the output wire is reachable from it. The **depth** of a wire w is the number of edges in the longest simple path from w to the output wire in the circuit graph. The **depth** of the circuit is the maximum depth of any relevant wire. The circuit is **acyclic** if the circuit graph contains no directed cycles. The circuit is **transitively reduced** if its circuit graph is transitively reduced, that is, if it contains no edge (u,w) such that there is a directed path of length at least two from u to w. In this paper we assume all circuits are acyclic.

2.2 Experiments

In an experiment some wires are constrained to be particular values or value distributions and the other wires are left free to take on values according to their gate functions and the values of their input wires. The behavior of a circuit consists of its responses to all possible experiments. For probabilistic circuits we consider both value injection experiments and distribution injection experiments.

A **distribution injection experiment** e is a function with domain W that maps each wire w to a special symbol $*$ or to a value distribution. A **value injection experiment** e is a distribution injection experiment for which every value distribution assigned is deterministic—that is, always generates the same symbol. To simplify notation, we think of a value injection experiment as a mapping from W to $(\Sigma \cup \{*\})$. If e is either kind of experiment, we say that e leaves w **free** if $e(w) = *$; otherwise we say that e **constrains** w to $e(w)$. If $e(w)$ is a single symbol, then we say e **fixes** w to $e(w)$.

We define a partial ordering $\leq$ on the set containing $*$ and all value distributions D as follows: $D \leq *$ for every value distribution D, and for two value distributions, $D_1 \leq D_2$ if the support of D_1 is a subset of the support of D_2. This ordering is extended to experiments on the same set of wires W as follows: $e_1 \leq e_2$ if for every $w \in W$, $e_1(w) \leq e_2(w)$. The intuitive meaning of $e_1 \leq e_2$ is that e_1 is at least as constraining as e_2 for every wire.

If e is any experiment, w is a wire, and a is $*$ or an element of Σ or a value distribution, then the experiment $e|_{w=a}$ is defined to be the experiment e' such that $e'(w) = a$ and $e'(u) = e(u)$ for all $u \in W$ such that $u \neq w$. If e is any experiment then a **free path** in e is a path in the circuit graph containing only wires w that are free in e.

2.3 Behavior

Let C be a probabilistic circuit. Then a distribution injection experiment e determines a joint distribution over assignments of elements of Σ to all of the wires of the circuit, as follows. If wire w

is constrained then w is randomly and independently assigned a value in Σ drawn according to the value distribution $e(w)$; in the case of a value injection experiment, this just assigns a fixed element of Σ to w. If wire w is free and has probabilistic gate function f, and its inputs $u_1, \ldots, u_k$ have been assigned the values $\sigma_1, \ldots, \sigma_k$, then w is randomly and independently assigned a value from Σ according to the value distribution determined by the gate function on these inputs, that is, according to the value distribution $f(\sigma_1, \ldots, \sigma_k)$.

Constrained gates and gates of fan-in zero give the base cases for the above recursive definition, which assigns an element of Σ to every wire because the circuit is acyclic. Let $C(e,w)$ denote the (marginal) value distribution of the assignments of values to w for the above process. The **output distribution** of the circuit, denoted $C(e)$, is the distribution $C(e,z)$, where z is the output wire of the circuit. The **behavior** of a circuit C is the function that maps value injection experiments e to output distributions $C(e)$.

We note that even when the circuit is Boolean and the only non-deterministic gates are uniform coin flips, the problem of exactly computing $C(e)$ is #**P**-hard because we can arrange for $C(e)$ to be the fraction of assignments satisfying a given Boolean formula.

2.4 Example: C_1

We give an example of a simple Boolean probabilistic circuit, which we also refer to later. The 2-input **averaging gate function** $A(b_1, b_2)$ outputs 1 with probability $(b_1 + b_2)/2$. Thus, if both inputs are 0, the output is deterministically 0, if both inputs are 1, the output is deterministically 1, and if its inputs disagree, the output is an unbiased coin flip, $U(\{0,1\})$. Another characterization of the averaging gate function A is that it randomly and equiprobably selects one of its inputs and copies it to the output.

We define a circuit C_1 of 4 wires as follows: $w_4 = A(w_2, w_3)$, $w_3 = w_1$, $w_2 = w_1$, and $w_1 = U(\{0,1\})$. The output wire is w_4. C_1 is depicted in Figure 1.

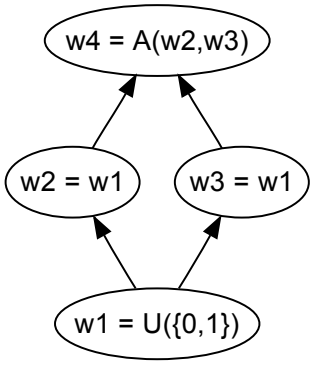


Figure 1: The circuit C_1; w_4 is the output wire.

To illustrate the behavior of this circuit, we consider two value injection experiments. Define the experiment e to leave every wire in C_1 free, that is, $e(w_i) = *$ for $1 \leq i \leq 4$. Given e, we construct one random outcome as follows. The wire w_1 is assigned a value as the result of an unbiased coin flip—say it is assigned 0. Then the values assigned to w_2 and w_3 are determined because they are each the output of an identity gate with w_1 as input: both are 0. Finally, because both its input wires have been assigned values, w_4 can be assigned a value according to $A(0,0)$, which is deterministically

0. It is easy to see that this is one of two possible outcomes for experiment e; either all wires are assigned 0 or all wires are assigned 1, and these each occur with probability $1/2$. The output distribution $C_1(e)$ is just an unbiased coin flip.

Now consider experiment $e' = e|_{w_2=1}$ that fixes w_2 to 1 and leaves the other wires free. Once again, the value of w_1 is determined by a coin flip—say it is assigned 0. Since w_2 is fixed to 1, that is its assignment. Wire w_3 is free, and is therefore assigned the value of w_1, that is 0. Now the inputs of w_4 have been assigned values, so we consider $A(1,0)$, which randomly and equiprobably selects 0 or 1. If, instead, the coin flip for w_1 had returned 1, all wires would be assigned 1. There are three possible assignments to (w_1,w_2,w_3,w_4) for experiment e': $(1,1,1,1)$ with probability $1/2$, $(0,1,0,0)$ with probability $1/4$ and $(0,1,0,1)$ with probability $1/4$. The output distribution $C_1(e')$ is a biased coin flip that is 1 with probability $3/4$.

2.5 Behavioral Equivalence

Two circuits C and C' are **behaviorally equivalent** if they have the same set of wires, the same output wire and the same behavior, that is, for every value injection experiment $e, C(e) = C'(e)$. We also need a concept of approximate equivalence. The **(statistical) distance** between value distributions D and D' is $d(D,D') = (1/2)\sum_\sigma |D(\sigma) - D'(\sigma)|$, which takes values in $[0,1]$. Note that when D and D' are deterministic, $d(D,D')$ is 0 if $D = D'$ and 1 otherwise. For $\varepsilon \geq 0, C$ is **ε-behaviorally equivalent** to C' if they contain the same wires and the same output wire, and for every value injection experiment e, $d(C(e),C'(e)) \leq \varepsilon$, where d is the statistical distance between value distributions.

In Lemma 2 we show that the behavioral equivalence of C and C' implies $C(e) = C'(e)$ for all distribution injection experiments as well. However, behavioral equivalence is not sufficient to guarantee that two circuits have the same topology; even when all the gates are Boolean, deterministic and relevant, the circuit graph of the target circuit may not be uniquely determined by its behavior (Angluin et al., 2009).

2.6 Queries

The learning algorithm gets information about the target circuit by specifying a value injection experiment e and observing the element of Σ assigned to the output wire. Such an action is termed a **value injection query**, abbreviated VIQ. A value injection query does not return complete information about the value distribution $C(e)$, but instead returns an element of Σ selected according to the distribution $C(e)$. Thus, in order to approximate the distribution $C(e)$, the learner must repeatedly make value injection queries with experiment e. In this case, the goal of learning is approximate behavioral equivalence.

2.7 The Learning Problem

The learning problem is ε-**approximate learning**: by making value injection queries to a target circuit C drawn from a known class of probabilistic circuits, the goal is to find a circuit C' that is ε-behaviorally equivalent to C. The inputs to the learning algorithm are the names of the wires in C, the name of the output wire and positive numbers ε and δ, where the learning algorithm is required to succeed with probability at least $(1-\delta)$.

We note that acyclic deterministic circuits are a subclass of acyclic probabilistic circuits. If the target circuit C is deterministic and we learn a probabilistic circuit C' that is $1/3$-behaviorally

equivalent to C, then we can compute the behavior of C on any value-injection experiment e with high probability by sampling the behavior of $C'(e)$. The negative results concerning learning deterministic circuits using value injection queries shown by Angluin et al. (2009) carry over to learning probabilistic circuits. In particular, for $\varepsilon = 1/3$ and $\delta = 1/2$, with no bound on fan-in or depth, the worst-case expected number of value injection queries necessary to learn acyclic probabilistic Boolean circuits is exponential, while with constant fan-in and no bound on depth, no polynomial time algorithm can learn acyclic probabilistic Boolean circuits if **NP** is not equal to **BPP**.

3. Preliminary Results

In this section we establish some basic results about probabilistic circuits, value injection experiments and distribution injection experiments. The reader may choose to skip this section and return to it as needed for proofs in subsequent sections.

We first note that if C is a probabilistic circuit, e is a distribution injection experiment and either $e(w)$ is a value distribution or e deterministically fixes all the input wires of w, then there is a value distribution D such that the value of w in $C(e)$ is determined by a random choice according to D, independent of the values chosen for any other wires. We make systematic use of this observation to reduce the number of experiments under consideration.

We start by considering two circuits C_1 and C_2 over the same wires, and distribution injection experiments e_1 and e_2 that agree on the distribution assigned to a wire w and that show a certain distance between $C_1(e_1)$ and $C_2(e_2)$. The following lemma says that we may modify e_1 and e_2 to fix w to a particular value $\sigma \in \Sigma$ while preserving (or increasing) the distance they show.

Lemma 1 *Let C_1 and C_2 be probabilistic circuits on wires W with the same output wire, let $w \in W$ be a wire, let D be a value distribution, and let e_1 and e_2 be distribution injection experiments such that $e_1(w) = e_2(w) = D$. Then there exists a value $\sigma \in \text{support}(D)$ such that*

$$d(C_1(e_1|_{w=\sigma}), C_2(e_2|_{w=\sigma})) \geq d(C_1(e_1), C_2(e_2)).$$

Proof We have

$$\begin{aligned}
d(C_1(e_1), C_2(e_2)) &= \frac{1}{2} \sum_{\tau \in \Sigma} \left| C_1(e_1)(\tau) - C_2(e_2)(\tau) \right| \\
&= \frac{1}{2} \sum_{\tau \in \Sigma} \left| \sum_{\rho \in \Sigma} C_1(e_1|_{w=\rho})(\tau) D(\rho) - \sum_{\rho \in \Sigma} C_2(e_2|_{w=\rho})(\tau) D(\rho) \right| \\
&\leq \frac{1}{2} \sum_{\rho \in \Sigma} D(\rho) \sum_{\tau \in \Sigma} \left| C_1(e_1|_{w=\rho})(\tau) - C_2(e_2|_{w=\rho})(\tau) \right| \\
&= \sum_{\rho \in \Sigma} D(\rho) d(C(e_1|_{w=\rho}), C(e_2|_{w=\rho})),
\end{aligned}$$

by the triangle inequality. Let

$$\sigma = \operatorname*{arg\,max}_{\rho \in \text{support}(D)} d(C(e_1|_{w=\rho}), C(e_2|_{w=\rho})),$$

so that

$$d(C(e_1|_{w=\sigma}), C(e_2|_{w=\sigma})) \geq d(C(e_1), C(e_2))$$

by an averaging argument. ■

By successively replacing each value distribution by a particular value, we may convert a distribution injection experiment that shows a certain distance between two circuits into a value injection experiment that shows at least that distance between the two circuits.

Lemma 2 *Let C_1 and C_2 be probabilistic circuits on wires W with the same output wire and let e be a distribution injection experiment. Then there exists a value injection experiment $e' \leq e$ such that*

$$d(C_1(e'),C_2(e')) \geq d(C_1(e),C_2(e)).$$

Proof By induction on $|V|$, where $V \subseteq W$ is the set of wires that e constrains to distributions that are not deterministic. If $|V| > 0$, then let $w \in V$. By Lemma 1, there exists a value $\sigma \in \Sigma$ such that

$$d(C_1(e|_{w=\sigma}),C_2(e|_{w=\sigma})) \geq d(C_1(e),C_2(e)).$$

Since $e|_{w=\sigma}$ constrains one fewer wire to a nonconstant distribution, the existence of e' follows from the inductive hypothesis. ■

Thus, value injection experiments suffice to establish approximate behavioral equivalence with respect to distribution injection experiments.

Corollary 3 *If circuits C_1 and C_2 are ε-behaviorally equivalent with respect to value injection experiments, then C_1 and C_2 are ε-behaviorally equivalent with respect to distribution injection experiments.*

Suppose that C is a probabilistic circuit and e_1 and e_2 are distribution injection experiments. For each wire w, we say that e_1 and e_2 **agree** on w if either

- e_1 and e_2 constrain w to the same distribution, or
- w is free in e_1 and e_2, and e_1 and e_2 agree on all of w's inputs.

It is clear that if e_1 and e_2 agree on a wire w, then the marginal distributions of w in e_1 and e_2 are identical, that is, $C(e_1,w) = C(e_2,w)$.

Lemma 4 *Let C be a probabilistic circuit on wires W and let e_1 and e_2 be distribution injection experiments that agree on wires $V \subseteq W$. Then there exist distribution injection experiments $e'_1 \leq e_1$ and $e'_2 \leq e_2$ such that for each wire $w \in V$, there exists a value $\sigma \in \Sigma$ such that $e'_1(w) = e'_2(w) = \sigma$, and*

$$d(C(e'_1),C(e'_2)) \geq d(C(e_1),C(e_2)).$$

Proof By induction on the number of unfixed wires $w \in V$. If there is such a wire, choose v by the acyclicity of the circuit to be one that is not reachable from the others. If $e_1(v) = e_2(v) = *$, then e_1 and e_2 agree on all of v's inputs, and by the choice of v, all of v's inputs are fixed. As

such, we may assume without loss of generality that e_1 and e_2 in fact constrain v to the distribution $D = C(e_1, v) = C(e_2, v)$. By Lemma 1, there exists a value $\sigma \in \text{support}(D)$ such that

$$d(C(e_1|_{v=\sigma}), C(e_2|_{v=\sigma})) \geq d(C(e_1), C(e_2)).$$

The existence of e_1' and e_2' follows from the inductive hypothesis. ■

The following lemma shows that constraining a wire w does not change the behavior of wires that are not reachable from w.

Lemma 5 *Let C be a probabilistic circuit on wires W, let e be a distribution injection experiment, let $w \in W$ be a wire free in e, and let D be a value distribution. Then e and $e|_{w=D}$ agree on all wires $u \in W$ such that there is no free path from w to u in e.*

Proof If u is constrained, then the conclusion follows. Otherwise, let $u \in W$ be a wire free in e such that there is no free path from w to u in e. Then no input v of u has a free path from w to v in e. We proceed by induction on the length of the longest path to u. If this length is zero, then u does not have any inputs. Otherwise, the inductive hypothesis applies to all of u's inputs, on which e and $e|_{w=D}$ then must agree. It follows that they also agree on u. ■

If we consider the distance between the behavior of a circuit with a wire constrained to two different value distributions, the following lemma allows us to move to a situation in which the wire is constrained to two different value distributions whose supports are disjoint. In the special case of Boolean circuits, the property of disjoint supports means that the resulting value distributions are deterministic. Later we see that this fundamentally distinguishes between alphabet size two and larger alphabets.

Lemma 6 *Let C be a probabilistic circuit on wires W, let $w \in W$ be a wire, and let D_1, D_2 be value distributions. There exist value distributions D_1', D_2' with $\text{support}(D_1') \cap \text{support}(D_2') = \emptyset$ such that for all experiments e,*

$$d(C(e|_{w=D_1}), C(e|_{w=D_2})) = d(D_1, D_2) d(C(e|_{w=D_1'}), C(e|_{w=D_2'})).$$

Proof Intuitively, we couple D_1 and D_2 so that $D_1 = D_2$ as often as possible and let $\widehat{D_i}$ be the distribution of D_i given that $D_1 \neq D_2$. It can be shown that $\widehat{D_1}$ and $\widehat{D_2}$ have disjoint support. Formally, we have

$$\begin{aligned} d(C(e|_{w=D_1}), C(e|_{w=D_2})) &= \frac{1}{2} \sum_{\sigma \in \Sigma} \left| C(e|_{w=D_1})(\sigma) - C(e|_{w=D_2})(\sigma) \right| \\ &= \frac{1}{2} \sum_{\sigma \in \Sigma} \left| \sum_{\tau \in \Sigma} C(e|_{w=\tau})(\sigma)(D_1(\tau) - D_2(\tau)) \right|. \end{aligned}$$

If we let

$$\begin{aligned} \widehat{D_1}(\tau) &= D_1(\tau) - \min(D_1(\tau), D_2(\tau)) \\ \widehat{D_2}(\tau) &= D_2(\tau) - \min(D_1(\tau), D_2(\tau)), \end{aligned}$$

then

$$d(C(e|_{w=D_1}),C(e|_{w=D_2})) = \frac{1}{2}\sum_{\sigma\in\Sigma}\left|\sum_{\tau\in\Sigma} C(e|_{w=\tau})(\sigma)(\widehat{D_1}(\tau)-\widehat{D_2}(\tau))\right|.$$

Since $\sum_{\tau\in\Sigma}\widehat{D_1}(\tau) = 1-\sum_{\tau\in\Sigma}\min(D_1(\tau),D_2(\tau))$ and likewise for D_2,

$$\begin{aligned} d(D_1,D_2) &= \frac{1}{2}\sum_{\tau\in\Sigma}\left|D_1(\tau)-D_2(\tau)\right| \\ &= \frac{1}{2}\sum_{\tau\in\Sigma}\left|\widehat{D_1}(\tau)-\widehat{D_2}(\tau)\right| \\ &= \sum_{\tau\in\Sigma}\widehat{D_1}(\tau) = \sum_{\tau\in\Sigma}\widehat{D_2}(\tau). \end{aligned}$$

If $d(D_1,D_2)>0$, then the distributions D_1' and D_2' where

$$\begin{aligned} D_1'(\tau) &= \widehat{D_1}(\tau)/d(D_1,D_2) \\ D_2'(\tau) &= \widehat{D_2}(\tau)/d(D_1,D_2) \end{aligned}$$

satisfy the requisite properties. Otherwise, any two distributions with disjoint support will do. ■

4. Test Paths

The concept of a test path has been central in previous work on learning deterministic circuits by means of value injection queries (Angluin et al., 2008b, 2009). A **test path** for a wire w, or **w-test path**, is a value injection experiment in which the free gates form a directed path in the circuit graph from w to the output wire. All the other wires in the circuit are fixed; this includes the inputs of w. A **side wire** with respect to a test path p is a wire fixed by p that is input to a free wire in p.

As an example, suppose that $\Sigma = \{0,1\}$ and the target circuit has a circuit graph as shown in Figure 2. There are four directed paths from w_1 to the output wire: w_1w_5, $w_1w_3w_5$, $w_1w_2w_4w_5$ and $w_1w_3w_4w_5$. A w_1-test path is a value injection experiment that sets the wires of one of these paths to $*$ and the other wires to 0 or 1, for example, $*011*$ or $**0**$. For the test path $*011*$, the side wires are w_3 and w_4, while for the test path $**0**$ the side wire is w_3. The value injection experiments $*****$ and $*01**$ are not test paths.

A test path may help the learning algorithm determine the effects of assigning different values to the wire w. The test path lemmas (Angluin et al., 2008b, 2009) may be re-stated as follows.

Lemma 7 *Let C be a deterministic circuit. If for some value injection experiment e, wire w free in e and alphabet symbols σ and τ it is the case that*

$$C(p|_{w=\sigma}) = C(p|_{w=\tau})$$

for every test path $p \le e$ then also

$$C(e|_{w=\sigma}) = C(e|_{w=\tau}).$$

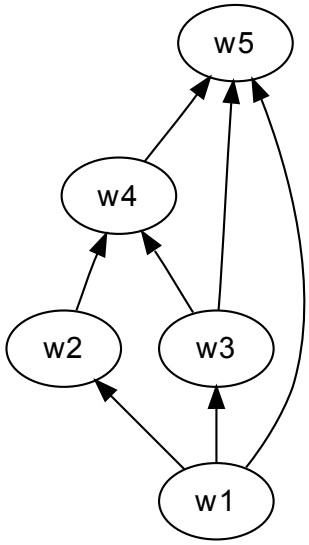


Figure 2: A circuit graph; w_5 is the output wire.

Nontrivial complications arise in attempting to carry over this test path lemma to general probabilistic circuits, as we now show. The following lemma shows that for alphabets of size at least three, there are transitively reduced probabilistic circuits for which the test-path lemma fails completely.

Lemma 8 *If $|\Sigma| \geq 3$, there exists a probabilistic circuit C, value injection experiment e, wire w free in e and alphabet symbols σ and τ such that although for every test path $p \leq e$ for w, $d(C(p|_{w=\sigma}), C(p|_{w=\tau})) = 0$, it is nevertheless the case that $d(C(e|_{w=\sigma}), C(e|_{w=\tau})) = 1/2$.*

Proof Assume that $\Sigma = \{0,1,2\}$, and define probabilistic gate functions T and X as follows.

$$\begin{aligned} T(0) &= T(1) = U(\{0,1\}) \\ T(2) &= 2 \\ X(b_1,b_2) &= b_1 \oplus b_2 \text{ if } b_1, b_2 \in \{0,1\} \\ X(b_1,b_2) &= U(\{0,1\}) \text{ if } b_1 = 2 \text{ or } b_2 = 2, \end{aligned}$$

where $\oplus$ is sum modulo 2. The gate function T flips a coin on input 0 or 1, and passes 2 through unaltered. The gate function X is exclusive or if neither input is 2, and a coin flip otherwise.

The circuit C has 5 wires, connected as in Figure 3. The output wire is w_5; note that C is transitively reduced.

Consider the experiment e that leaves all the wires free. In this experiment, we have $C(e|_{w_1=0}) = C(e|_{w_1=1}) = 0$ because w_2 is a coin flip and w_5 is the exclusive or of two copies of the coin flip. On the other hand, $C(e|_{w_1=2}) = U(\{0,1\})$ because $w_4 = w_3 = w_2 = 2$ and w_5 is therefore a coin flip. Thus $d(C(e|_{w_1=0}), C(e|_{w_1=2})) = 1/2$.

However, the only test paths for w_1 fix w_3 and leave all other wires free, or fix w_4 and leave all other wires free, and the two cases are symmetric. If w_3 is fixed to any value and all other wires are free, then w_5 is a coin flip when $w_1 = 2$. If w_3 is fixed to 2 and all other wires are free, then w_5 is also a coin flip. If w_3 is fixed to $b \in \{0,1\}$ and all other wires are free, then when $w_1 \in \{0,1\}$, w_2 is a coin

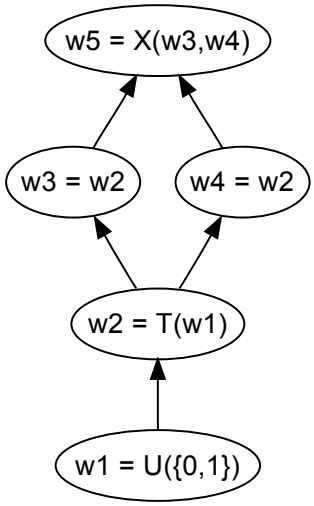


Figure 3: The circuit C; w_5 is the output wire.

flip, and w_5 is the exclusive or of b and that coin flip, that is, w_5 is also coin flip. Hence, for any test path $p \leq e$ for w_1, we have $C(p|_{w_1=0}) = C(p|_{w_1=2}) = U(\{0,1\})$ and $d(C(p_{w_1=0}), C(p_{w_1=2})) = 0$.

For alphabets Σ of size larger than 3, we can treat three of the symbols as 0, 1 and 2 in the above construction, and the other symbols as "tilt," where each function outputs a tilt value if any of its inputs is a tilt value. ■

4.1 A Bound for Boolean Probabilistic Circuits

Surprisingly, the case of $|\Sigma| = 2$ is different; for Boolean probabilistic circuits there is a useful quantitative relationship between the difference exposed by an arbitrary experiment e and the differences exposed by test paths $p \leq e$. The bound we give depends on the structure of directed paths on free wires in e.

Let e be an experiment and w a wire. Define $\Pi(e,w)$ to be the set of all directed paths from w to the output wire on free wires in e. Let $S(e)$ be the set of wires that originate a free shortcut, that is, the set of free wires w such that there exists a path $p \in \Pi(e,w)$ with two free wires to which w is an input. Define

$$\kappa(e,w) = \sum_{p \in \Pi(e,w)} 2^{|p \cap S(e)|}.$$

Thus, $\kappa(e,w)$ is the sum over paths in $\Pi(e,w)$ of 2 raised to the number of wires on the path that originate free shortcuts in e. If there are no wires that originate free shortcuts in e, then this is just the number of free paths in e. As an example, if the target circuit has the circuit graph shown in Figure 2 and the experiment e leaves all wires free then $\Pi(e,w_1)$ contains the four paths w_1w_5, $w_1w_3w_5$, $w_1w_2w_4w_5$ and $w_1w_3w_4w_5$, $S(e) = \{w_1, w_3\}$, and $\kappa(e,w)$ is $2+4+2+4 = 12$.

The following technical lemma gives a useful recurrence for $\kappa(e,w)$.

Lemma 9 *Let C be a probabilistic circuit, e be a distribution injection experiment, w and u be free wires where w is an input to u, and D_0 be a value distribution. Let $\beta = 2$ if $w \in S(e)$ and $\beta = 1$*

otherwise. Then

$$\kappa(e,w) = \kappa(e|_{u=D_0},w) + \kappa(e|_{w=1},u)\cdot\beta.$$

Proof The first term of the sum counts paths that don't contain u, and the second counts paths that do. Let $e' = e|_{u=D_0}$ and $e'' = e|_{w=1}$. We have

$$\begin{aligned}
\kappa(e,w) &= \sum_{p\in\Pi(e,w)} 2^{|p\cap S(e)|} \\
&= \sum_{\substack{p\in\Pi(e,w)\\ u\notin p}} 2^{|p\cap S(e)|} + \sum_{\substack{p\in\Pi(e,w)\\ u\in p}} 2^{|p\cap S(e)|} \\
&= \sum_{p\in\Pi(e',w)} 2^{|p\cap S(e')|} + \sum_{p\in\Pi(e'',u)} 2^{|p\cap S(e'')|}\beta \\
&= \kappa(e',w) + \kappa(e'',u)\cdot\beta,
\end{aligned}$$

since each path $p \ni u$ from w corresponds to the path $p\setminus\{w\}$ from u. ■

Next is the key lemma relating the difference exposed by e to the differences exposed by paths $p \le e$ for Boolean probabilistic circuits.

Lemma 10 *Let C be a Boolean probabilistic circuit, e be a distribution injection experiment, w be a wire free in e and D_1, D_2 be value distributions. If there exists $\varepsilon \ge 0$ such that for all w-test paths $p \le e$,*

$$d(C(p|_{w=D_1}), C(p|_{w=D_2})) \le \varepsilon,$$

then

$$d(C(e|_{w=D_1}), C(e|_{w=D_2})) \le \kappa(e,w)\cdot\varepsilon.$$

Proof By induction on $\phi(e)$, the number of free wires in e. By Lemma 6, assume that $\text{support}(D_1)\cap\text{support}(D_2) = \emptyset$. The critical feature of the Boolean case is that it follows that $D_1 = 0$ and $D_2 = 1$ without loss of generality—it is important to the following proof that D_1 and D_2 be deterministic.

If $\phi(e) = 1$, then either

$$d(C(e|_{w=0}), C(e|_{w=1})) = 0,$$

or w is the output, e is a w-test path, and $\kappa(e,w) = 1$. Otherwise, the inductive hypothesis is that the lemma holds for all experiments e' with $\phi(e') < \phi(e)$.

Except for w, the experiments $e|_{w=0}$ and $e|_{w=1}$ agree on all constrained wires, so by Lemmas 4 and 5, assume without loss of generality that every wire with no free path from w is in fact fixed. Since C is acyclic, there exists a free wire $u \ne w$ whose only unfixed input is w. Let g be the gate assigned by C to u and let $B_0 = g(e|_{w=0})$ and $B_1 = g(e|_{w=1})$, so that

$$\begin{aligned}
C(e|_{w=0}) &= C(e|_{w=0,u=B_0}) \\
C(e|_{w=1}) &= C(e|_{w=1,u=B_1}).
\end{aligned}$$

By the triangle inequality,

$$\begin{aligned}
d(C(e|_{w=0}), C(e|_{w=1})) \le\ & d(C(e|_{w=0,u=B_0}), C(e|_{w=1,u=B_0})) \\
& + d(C(e|_{w=1,u=B_0}), C(e|_{w=1,u=B_1})).
\end{aligned}$$

Letting $e' = e|_{u=B_0}$, any test path $p \leq e'$ also satisfies $p \leq e$ since $e' \leq e$. The experiment e' has one fewer free wire, as u is free in e, so using the inductive hypothesis, we can bound the first term of the sum by $\kappa(e', w) \cdot \varepsilon$. We now derive a bound on u-test paths so that the inductive hypothesis applies to the second term as well. Let $\beta = 2$ if $w \in S(e)$ and $\beta = 1$ otherwise. Let $e'' = e|_{w=1}$ and suppose $p \leq e''$ is a u-test path. Then

$$\begin{aligned}
&d(C(p|_{u=B_0}), C(p|_{u=B_1})) \\
&\leq d(C(p|_{w=1,u=B_0}), C(p|_{w=0,u=B_0})) + d(C(p|_{w=0,u=B_0}), C(p|_{w=1,u=B_1})) \\
&\text{[by the triangle inequality]} \\
&= d(C(p|_{w=1,u=B_0}), C(p|_{w=0,u=B_0})) + d(C(p|_{w=0,u=*}), C(p|_{w=1,u=*})) \\
&\text{[by the definitions of } B_0 \text{ and } B_1\text{]}.
\end{aligned}$$

Since w is an input to u, both $p|_{w=*,u=B_0}$ and $p|_{w=*,u=*}$ are w-test paths. Therefore, both terms of the sum are bounded by ε, and the first is nonzero only if w is an input to some free wire in p other than u. It follows that

$$d(C(p|_{u=B_0}), C(p|_{u=B_1})) \leq \beta\varepsilon,$$

and thus that

$$d(C(e''|_{u=0}), C(e''|_{u=1})) \leq \kappa(e'', u) \cdot \beta\varepsilon,$$

so by Lemma 9,

$$\begin{aligned}
d(C(e|_{w=0}), C(e|_{w=1})) &\leq \kappa(e', w) \cdot \varepsilon + \kappa(e'', u) \cdot \beta\varepsilon \\
&= \kappa(e, w) \cdot \varepsilon.
\end{aligned}$$

■

In the case of transitively reduced circuits, $S(e) = \emptyset$, and $\kappa(e, w) = \pi(e, w)$, where $\pi(e, w) = |\Pi(e, w)|$, the number of directed paths on free wires in e from w to the output wire.

Corollary 11 *Let C be a transitively reduced Boolean probabilistic circuit, e be a distribution injection experiment, and w be a wire free in e. If there exists $\varepsilon \geq 0$ such that for all w-test paths $p \leq e$,*

$$d(C(p|_{w=0}), C(p|_{w=1})) \leq \varepsilon,$$

then

$$d(C(e|_{w=0}), C(e|_{w=1})) \leq \pi(e, w) \cdot \varepsilon.$$

5. Learning Boolean Probabilistic Circuits

The amount of attenuation given by Lemma 10 allows us to adapt the Circuit Builder algorithm (Angluin et al., 2009) to learn Boolean probabilistic circuits with constant fan-in and log depth in polynomial time. For this class of circuits, the attenuation factor $\kappa(e, w)$ is bounded by a polynomial in n.

Theorem 12 *Given constants c and k there is a nonadaptive learning algorithm that with probability at least $(1-\delta)$ successfully ε-approximately learns any Boolean probabilistic circuit with n wires, gates of fan-in at most k and depth at most $c\log n$ using value injection queries in time bounded by a polynomial in n, $1/\varepsilon$ and $\log(1/\delta)$.*

The rest of the section is devoted to proving this theorem. Let the target circuit be C with $\Sigma = \{0,1\}$ and let positive constants δ, ε, k and c be given such that the fan-in of C is bounded by k and the depth of C is bounded by $c\log n$. For such a circuit, $\pi(e,w)$ is bounded above by $k^{c\log n}$, so the quantity $\kappa(e,w)$ is bounded above by

$$\kappa(n) = k^{c\log n} \cdot 2^{c\log n} = n^{c(\log k+1)} = n^{O(1)}.$$

We now describe our Probabilistic Circuit Builder algorithm (PCB). PCB is nonadaptive: first it computes a set U of value injection experiments such that every test path is equivalent to some experiment in U. It then repeats each value injection query $e \in U$ enough times that with probability at least $(1-\delta)$, the distribution $C(e)$ is estimated with sufficient accuracy for every $e \in U$. Finally, it uses these estimates to build a circuit C' by repeatedly adding a sufficiently accurate gate all of whose inputs are in the partially constructed circuit. If the estimates of $C(e)$ are all sufficiently accurate, then C' is ε-behaviorally equivalent to C.

5.1 Constructing U

In choosing the experiments U, the goal is that for every potential test path, U includes an equivalent experiment. The structure of the circuit, however, is not known *a priori*, a difficulty that we overcome by the same method as used by Angluin et al. (2009). Let U_* be a universal set of value injection experiments such that for every set of $kc\log n$ wires and every assignment of symbols from $\Sigma \cup \{*\}$ to those wires, some experiment $e \in U_*$ agrees with the values assigned to those wires. There is a deterministic construction of such a set U_* of size

$$2^{O(kc\log n)}\log n = n^{O(kc)}$$

in time polynomial in its size (Angluin et al., 2009). (For intuition, a set of $n^{O(kc)}$ independent random uniform assignments of $*$, 0 and 1 to the wires has this property with high probability.) For every wire w and test path p for w, there is an experiment in U_* that leaves the path wires of p free and fixes the side wires of p to their values in p. Consequently, p and this experiment agree on the output wire. In order to have experiments in which each free wire is also set to 0 and 1, for $b = 0,1$ let U_b contain every experiment $e|_{w=b}$ such that $e \in U_*$ and w is free in e. The final set of experiments is $U = U_* \cup U_0 \cup U_1$.

5.2 Estimating $C(e)$ for $e \in U$

For each $e \in U$, PCB repeatedly makes a value injection query with e to estimate the value distribution $C(e)$; let $\widehat{C}(e)$ denote this estimate. By Hoeffding's bound, we have that

$$m = O((n\kappa(n)/\varepsilon)^2 \log(|U|/\delta))$$

trials per experiment e suffice to guarantee that with probability at least $1-\delta$, for all $e \in U$,

$$d(C(e), \widehat{C}(e)) \leq \varepsilon/(4n\kappa(n)). \tag{1}$$

Let $e \in U_*$ be a value injection experiment, w be a wire that e leaves free, and D be a value distribution. We define

$$\widehat{C}(e|_{w=D}) = \sum_{\sigma \in \Sigma} D(\sigma)\widehat{C}(e|_{w=\sigma}).$$

Note that this is computed from the values of $\widehat{C}(e|_{w=\sigma})$ and does not require new experiments.

If (1) holds for all $e \in U$, then we have

$$\begin{aligned} d(C(e|_{w=D}), \widehat{C}(e|_{w=D})) &\leq \sum_{\sigma \in \Sigma} D(\sigma) d(C(e|_{w=\sigma}), \widehat{C}(e|_{w=\sigma})) \\ &\leq \varepsilon/(4n\kappa(n)). \end{aligned} \tag{2}$$

5.3 Building the Circuit C'

PCB builds the circuit C' one gate at a time. Let W' denote the set of wires of C' that have already been assigned a gate by PCB; initially W' is empty. While $W' \neq W$, PCB attempts to add another gate to C' by searching for a wire $w \in (W - W')$ and a probabilistic gate g' all of whose inputs are in W' such that for each experiment $e \in U_*$ that leaves w free and fixes all inputs of g',

$$d(\widehat{C}(e), \widehat{C}(e|_{w=g'(e)})) \leq 2\varepsilon/(4n\kappa(n)).$$

If no such gate can be found or $W' = W$, PCB outputs C' and halts. We will later show that a gate can be found as long as $W \neq W'$.

The search for g' iterates over every wire $w \in (W - W')$ and every choice of an r-tuple of distinct wires $w_1, \ldots, w_r$ from W' as the inputs of w, where $0 \leq r \leq k$. For each such choice, PCB attempts to define a probabilistic gate function f as follows. For each $(\sigma_1, \ldots, \sigma_r) \in \Sigma^r$, PCB seeks a number $x \in [0,1]$ such that if D_x is the distribution that is 1 with probability x and 0 with probability $(1-x)$ then

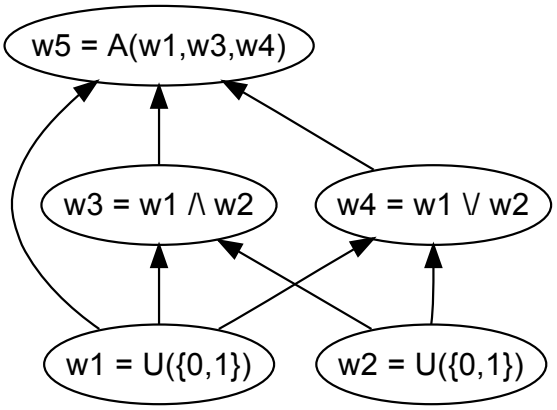
$$d(\widehat{C}(e), \widehat{C}(e|_{w=D_x})) \leq 2\varepsilon/(4n\kappa(n))$$

for all experiments $e \in U_*$ that leave w free and fix w_i to σ_i for $i = 1, \ldots, r$. Since the left hand side is a convex function of x, every such e constrains the possible values of x to an interval, and any x in the intersection of $[0,1]$ and the intervals for all such e suffices. If the intersection is empty, then the attempt to define f fails; otherwise, $f(\sigma_1, \ldots, \sigma_r)$ is defined to be D_x. If PCB succeeds in defining f for all possible r-tuples $(\sigma_1, \ldots, \sigma_r)$, then the gate g' with inputs $w_1, \ldots, w_r$ and probabilistic gate function f is assigned to w.

5.4 An Illustration

For some intuition about the operation of PCB, consider the probabilistic Boolean circuit shown in Figure 4. Wires w_1 and w_2 are determined by random coin flips, w_3 is the AND of w_1 and w_2, w_4 is the OR of w_1 and w_2, and w_5 is determined by the 3-input averaging gate applied to w_1, w_3 and w_4. The table shows the probability that $w_5 = 1$ for a selected set of value injection experiments.

Suppose that these experiments are contained in U when PCB attempts to add the first gate to C'. Of course, PCB will only have repeated sampling estimates of these probabilities, but suppose for a moment that the exact values were available. Because W' is empty, the first gate added must



#	Experiment	$\Pr[w_5 = 1]$
1.	* * * * *	1/2
2.	0 * * * *	1/6
3.	1 * * * *	5/6
4.	* * 0 * *	5/12
5.	* * 1 * *	3/4
6.	* 1 * 0 *	1/3
7.	0 1 * 0 *	0
8.	1 1 * 0 *	2/3
9.	* 1 0 0 *	1/6
10.	* 1 1 0 *	1/2

Figure 4: A Boolean circuit with output wire w_5, and some of its behavior.

have no inputs and must be determined by a coin flip that is 1 with some probability x. In this group of experiments, there are two constraints for wire w_1 for the possible values of x. Experiments 1, 2 and 3 give the constraint $(1/6)(1-x)+(5/6)x = 1/2$, which implies $x = 1/2$, and experiments 6, 7 and 8 give the constraint $0(1-x)+(2/3)x = 1/3$, which also implies $x = 1/2$, consistent with the gate computing w_1 in the target circuit. There are also two constraints on the possible values of x for the wire w_3. Experiments 1, 4 and 5 give the constraint $(5/12)(1-x)+(3/4)x = 1/2$, which implies $x = 1/4$, and experiments 6, 9 and 10 give the constraint $(1/6)(1-x)+(1/2)x = 1/3$, which implies $x = 1/2$. Thus there is no consistent value of x that would allow the first gate to be chosen for wire w_3. Rather than exact values, PCB considers intervals determined by error tolerances, but when these are small enough, the constraint intervals for w_3 will not overlap, and PCB will not choose the first gate for wire w_3.

5.5 Correctness

With probability at least $(1-\delta)$, the estimates $\widehat{C}(e)$ satisfy (1) for all $e \in U$. We now assume that the estimates satisfy these bounds and show that PCB successfully builds a circuit C' that is ε-behaviorally equivalent to C.

We first establish two lemmas connecting gates, paths and experiments. Given a Boolean probabilistic circuit C and a probabilistic gate g, g is **η-correct** for wire w with respect to C if for every value injection experiment e that fixes the input wires for g we have $d(C(e), C(e|_{w=g(e)})) \leq \eta$, where $g(e)$ denotes the value distribution determined by g when its inputs are fixed as in e. Recall that $\phi(e)$ denotes the number of free wires in experiment e, and therefore $\phi(e) \leq n$ for all e.

Lemma 13 *Let C and C' be probabilistic circuits on wires W, and let e be a distribution injection experiment. If for every wire w, the gate for w in C' is η-correct for w with respect to C, then*

$$d(C(e), C'(e)) \leq \phi(e) \cdot \eta.$$

Proof By induction on $\phi(e)$, the number of free wires in e. If $\phi(e) = 0$, then e constrains the output wire, and trivially, $d(C(e), C'(e)) = 0$. Otherwise, the inductive hypothesis is that

$$d(C(e'), C'(e')) \leq \phi(e') \cdot \eta$$

for all experiments e' with fewer than $\phi(e)$ free gates.

By Lemma 2, assume that e is in fact a value injection experiment. Since C' is acyclic, there exists a free wire w in e such that the inputs to w in C' are fixed in e to some k-tuple $(\sigma_1,\ldots,\sigma_k) \in \Sigma^k$. Let f denote the probabilistic gate function for w in C', and let D denote the value distribution $f(\sigma_1,\ldots,\sigma_k)$. Then we have $C'(e) = C'(e|_{w=D})$, and

$$\begin{aligned} d(C(e),C'(e)) &\le d(C(e),C(e|_{w=D})) + d(C(e|_{w=D}),C'(e|_{w=D})) \\ &\le \eta + (\phi(e)-1)\cdot\eta \\ &= \phi(e)\cdot\eta \end{aligned}$$

by the inductive hypothesis and the fact that f is η-correct for w. ■

Corollary 14 *Let C and C' be probabilistic circuits on wires W where $|W| = n$. If for every wire w, the gate g for w in C' is η-correct for w with respect to C, then*

$$d(C(e),C'(e)) \le n\cdot\eta.$$

Proof By the definition of approximate behavioral equivalence and the bound $\phi(e) \le n$. ■

Next we show that test paths are sufficient to determine whether a gate is η-correct for a wire in C.

Lemma 15 *Let C be a Boolean probabilistic circuit, w a wire and g' a probabilistic gate. If for every test path p for w that fixes all the inputs of g', $d(C(p),C(p|_{w=g'(p)})) \le \eta/K_w$, where K_w is the maximum value of $\kappa(e,w)$ for C over all experiments e, then g' is η-correct for w with respect to C.*

Proof Let g be the actual gate that C assigns to w. Let e be a value injection experiment that fixes every input of g'. Then e may not fix all of g's inputs, but because C is acyclic, g's inputs are not reachable from w. By Lemmas 4 and 5, there exists an experiment $e' \le e$ that fixes g's inputs, with

$$d(C(e'),C(e'|_{w=g'(e')})) \ge d(C(e),C(e|_{w=g'(e)})).$$

Since e' fixes all of g's inputs, $C(e') = C(e'|_{w=g(e')})$. It is given that for all test paths p that fix all inputs of g' that

$$d(C(p|_{w=g(p)}),C(p|_{w=g'(p)})) \le \eta/K_w,$$

so it follows by Lemma 10 that

$$d(C(e'|_{w=g(e')}),C(e'|_{w=g'(e')})) \le \kappa(e',w)\cdot\eta/K_w \le \eta,$$

and g' is η-correct for w. ■

To prove that PCB constructs a circuit C' that is ε-behaviorally equivalent to the target circuit C, we show that for each wire $w \in W$, PCB assigns a gate that is ε/n-correct for w in C.

Assume that $W' \neq W$, that is, that not all wires have been assigned gates, and consider PCB as it attempts to add another gate to C'. PCB looks for a wire $w \in (W - W')$ and probabilistic gate $g' \in G$ with all of its inputs in W' such that for each experiment $e \in U_*$ that leaves w free and fixes all inputs of g',

$$d(\widehat{C}(e), \widehat{C}(e|_{w=g'(e)})) \leq 2\varepsilon/(4n\kappa(n)).$$

If this search succeeds, then by (1),

$$d(C(e), \widehat{C}(e)) \leq \varepsilon/(4n\kappa(n)) d(\widehat{C}(e|_{w=g'(e)}), C(e|_{w=g'(e)})) \leq \varepsilon/(4n\kappa(n)),$$

and thus by the triangle inequality we have

$$d(C(e|_{w=g'(e)}), C(e)) \leq \varepsilon/(n\kappa(n)),$$

It follows by Lemma 15 and the choice of $\kappa(n)$ that g' is ε/n-correct for w in C.

To see that the search for a gate will succeed as long as $W' \neq W$, we note that because C is acyclic, there is some wire $w \in (W - W')$ such that all of w's inputs in C are in W'. Let g denote the gate assigned by C to w, with inputs $w_1, \ldots, w_r$ and probabilistic gate function f. By the existence of g, there is at least one feasible gate-wire assignment for PCB to make, ensuring the continued progress of PCB. Consider any experiment $e \in U_*$ that leaves w free and fixes the inputs of g to $(\sigma_1, \ldots, \sigma_r)$. Let D be the value distribution $f(\sigma_1, \ldots, \sigma_r)$. Then $C(e) = C(e|_{w=D})$ and by (1) and (2) we have

$$\begin{aligned} d(\widehat{C}(e), C(e)) &\leq \varepsilon/(4n\kappa(n)) \\ d(C(e|_{w=D}), \widehat{C}(e|_{w=D})) &\leq \varepsilon/(4n\kappa(n)), \end{aligned}$$

so by the triangle inequality,

$$d(\widehat{C}(e), \widehat{C}(e|_{w=D})) \leq 2\varepsilon/(4n\kappa(n)).$$

Therefore, PCB will continue to make progress until it has assigned a gate to every wire in W, and every such gate will be ε/n-correct for its wire in C, which means that C' will be ε-behaviorally equivalent to C.

5.6 Running Time

To bound the running time of PCB we argue as follows. The set U of experiments is of cardinality $n^{O(kc)}$ and can be constructed in time polynomial in its size. To estimate $C(e)$, each experiment in U is repeated

$$O((n\kappa(n)/\varepsilon)^2 \log(|U|/\delta))$$

times; recall that $\kappa(n) = O(n^{c(\log k+1)})$. PCB then chooses a gate for a wire n times. For each choice, it must at worst iterate over $O(n)$ wires in $(W - W')$, over all $O(n^k)$ choices of k or fewer input wires from W', over all $|\Sigma|^k$ assignments of values to the input wires, and all experiments in U. Thus the running time of PCB is polynomial in n, $1/\varepsilon$ and $1/\delta$.

6. Lower Bounds on Path Attenuation

The path attenuation bound $\kappa(n)$ is a significant factor in the running time of the PCB algorithm. In this section we consider lower bounds on path attenuation for Boolean probabilistic circuits. The following theorem shows that the bound of $\pi(e,w)$ for transitively reduced Boolean probabilistic circuits in Corollary 11 is tight infinitely often.

Theorem 16 *There is an infinite set of transitively reduced probabilistic Boolean circuits such that for each circuit C in the family, there exists a value injection experiment e and a wire w free in e such that*

$$d(C(e|_{w=0}),C(e_{w=1})) = 1$$

and for every test path p for w we have

$$d(C(p|_{w=0}),C(p|_{w=1})) = 1/\pi(e,w).$$

Proof For each positive integer ℓ, define the circuit C_ℓ to be a chain of ℓ copies of the circuit C_1 in Figure 1 with wire w_4 of one copy identified with wire w_1 of the next copy. More formally, the $3\ell+1$ wires are $w_{0,4}$ and $w_{i,j}$ for $i = 1,\ldots,\ell$ and $j = 2,3,4$. The output wire is $w_{\ell,4}$. The wire $w_{0,4}$ has no inputs and is determined by an unbiased coin flip, that is, $U(\{0,1\})$. The wires $w_{i,2}$ and $w_{i,3}$ are the outputs of deterministic identity gates with input $w_{i-1,4}$. The wire $w_{i,4} = A(w_{i,2},w_{i,3})$ is the result of applying the two-input averaging probabilistic gate function A to the wires $w_{i,2}$ and $w_{i,3}$. The circuit C_3 is depicted in Figure 5.

To understand the operation of this circuit in response to a value injection experiment e, we may view each averaging gate as choosing one of its inputs to copy to its output. Starting at the output wire, this determines a path back to the first wire whose value has been fixed, or to the wire $w_{0,4}$ (which has no inputs) and the output of the circuit is the value of the wire so reached.

Define experiment e to leave all of the wires free. Let w denote the wire $w_{0,4}$. Clearly there are 2^ℓ paths on free gates in e from w to the output gate, that is, $\pi(w,e) = 2^\ell$. For experiment e every possible path starts at wire w and we have $C(e|_{w=0}) = 0$ and $C(e|_{w=1}) = 1$, so $d(C(e|_{w=0}),C(e|_{w=1})) = 1$. However, any test path p for w must fix one of the wires $w_{i,2}$ or $w_{i,3}$ for each $i = 1,\ldots,\ell$. Thus, there is exactly one path that leads back to wire w, and this path is the one chosen by the averaging gates with probability $1/2^\ell$. Thus the result for any test path p for w is $d(C(p|_{w=0}),C(p|_{w=1})) = 1/2^\ell = 1/\pi(e,w)$. ■

This lower bound also holds for general transitively reduced circuit topologies, as follows. (Note that this result was incorrectly stated in the preliminary version of this paper (Angluin et al., 2008a).)

Theorem 17 *Let G be a transitively reduced acyclic directed graph with a designated output node z that is reachable from every node. For each node w there exists a Boolean probabilistic circuit C whose circuit graph is G with output wire z such that for every value injection experiment e that leaves w free and for every test path $p \leq e$ for wire w we have*

$$d(C(e|_{w=1}),C(e|_{w=0})) \geq \pi(e,w) \cdot d(C(p|_{w=1}),C(p|_{w=0})).$$

Proof Let w be given. To construct C, each node v of G is assigned a probabilistic gate whose inputs are the in-neighbors of v in G, as follows. For each node v, let $P(v)$ denote the number of

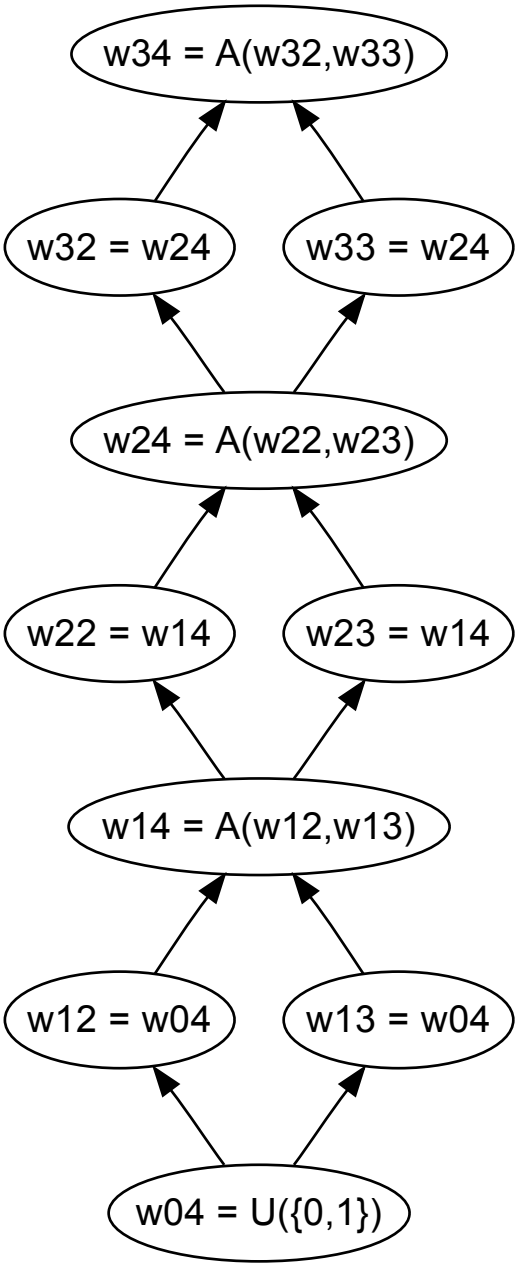


Figure 5: The circuit C_3; $w_{3,4}$ is the output wire.

distinct directed paths from w to z that include node v, and for each edge (u,v), let $P(u,v)$ denote the number of distinct directed paths from w to z that include edge (u,v). If there are no paths from w to z through v (that is, $P(v) = 0$) then we let the probabilistic gate function for v be the constant function 0. The probabilistic gate function for w is a coin flip, $U(\{0,1\})$.

Otherwise, if node v has inputs $u_1, \ldots, u_r$ then it is assigned the probabilistic gate function specified by

$$A_v(b_1, \ldots, b_r) = \sum_{i=1}^{r} b_i \cdot P(u_i, v)/P(v)$$

This generalizes the two-input averaging gate A, weighting input u_i by the fraction of paths from w to z passing through v that also pass through u_i. We may view A_v as performing a random weighted selection of one of its inputs to copy to its output. The weights have been chosen so that each directed path from w to z is selected with probability $1/P(w)$.

Let e be any value injection experiment that leaves w free. If there is no path on free wires in e from w to the output, then $\pi(e,w) = 0$, and the bound in the conclusion of the lemma holds trivially. Otherwise, the output of the circuit in response to e is determined by tracing from the output wire, following the choices of the averaging gates, until either the first wire fixed by e, or w, is reached. Thus

$$d(C(e|_{w=1}), C(e|_{w=0})) = \pi(e,w)/P(w),$$

because there are $\pi(e,w)$ paths from w to the output wire in e. Let $p \leq e$ be any test path for w; now there is just one choice of path that leads back to w, so

$$d(C(p|_{w=1}), C(p|_{w=0})) = 1/P(w),$$

establishing the conclusion of the lemma. ■

Can the general bound in Lemma 10 be improved to the bound for transitively reduced circuits in Corollary 11? The following example shows that the better bound is in general not attainable if the circuit is not transitively reduced. It gives a family of circuits of depth 2ℓ for which the worst-case ratio of the differences shown for w by an experiment e and the best path for w is $(5/4)^\ell \pi(e,w)$.

Theorem 18 *There exists an infinite set of Boolean probabilistic circuits $D_1, D_2, \ldots$ such that for each ℓ there exists a value injection experiment e and a wire w free in e such that $\pi(e,w) = 4^\ell$ and*

$$d(D_\ell(e|_{w=0}), D_\ell(e|_{w=1})) = (5/7)^\ell,$$

but for any test path p for w,

$$d(D_\ell(p|_{w=0}), D_\ell(p|_{w=1})) = (1/7)^\ell.$$

Proof We first define a Boolean probabilistic circuit D_1 and then connect ℓ copies of it in series to get D_ℓ. The wires of D_1 are $w_1, \ldots, w_5$. They are connected as in Figure 6; the output wire is w_5. Note that the edge (w_1, w_5) means that the circuit graph is not transitively reduced. The gate

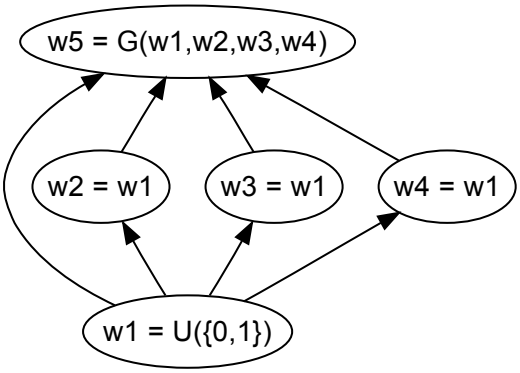


Figure 6: The circuit D_1; w_5 is the output wire.

function G is defined by giving its expected value as a function of its inputs:

$$E[G(w_1,w_2,w_3,w_4)] = ((1-w_1)+2w_2+2w_3+2w_4)/7.$$

Let e be the experiment that leaves all five wires free. It is clear that

$$d(D_1(e|_{w_1=0}),D_1(e|_{w_1=1})) = 5/7.$$

We now show that for any test path p for w_1,

$$d(D_1(p|_{w_1=0}),D_1(p|_{w_1=1})) = 1/7.$$

The possible test paths p for w_1 either fix all of w_2,w_3,w_4 or all but one of them. Thus, as we change from $w_1 = 0$ to $w_1 = 1$ in such a test path, the assignments to wires (w_1,w_2,w_3,w_4) change in one of four possible ways:

$(0,b_2,b_3,b_4)$ to $(1,b_2,b_3,b_4)$

$(0,0,b_3,b_4)$ to $(1,1,b_3,b_4)$

$(0,b_2,0,b_4)$ to $(1,b_2,1,b_4)$

$(0,b_2,b_3,0)$ to $(1,b_2,b_3,1)$

Checking each of these possible changes against the definition of G, we see that each change produces a difference of $1/7$, as claimed. (This example can be modified to give a difference of 1 versus $1/5$.) Thus, setting $w = w_1$, the circuit D_1 gives the base case of the claim in the lemma.

To construct D_ℓ, we take ℓ copies of D_1 and identify wire w_5 in one copy with wire w_1 in the next copy, making the wire w_5 of the final copy the output wire of the whole circuit. Let w denote the wire w_1 in the first such copy. Then $\pi(e,w) = 4^\ell$ and

$$d(D_\ell(e|_{w=0}),D_\ell(e_{w=1})) = (5/7)^\ell.$$

For any test path p, the signal is attenuated by a factor of $1/7$ for each level, and we have

$$d(D_\ell(p|_{w=0}),D_\ell(p|_{w=1})) = 1/7^\ell.$$

■

This construction can be generalized to $k+1$ wires for any odd $k+1$, which increases the attenuation. In the base circuit there are k paths and an attenuation factor of $1/(2k-3)$, and the worst-case ratio of differences for an experiment and its test paths in D_ℓ approaches $2^\ell \pi(e,w)$ as k goes to infinity.

7. Exponential Dependence on Depth

The bounds on path attenuation show that test paths may be much less informative than general value injection experiments, resulting in the exponential dependence of the number of experiments and the running time of PCB on the depth of the target circuit. It is natural to ask whether we might do better by using selected general experiments. In this section, we give computational evidence to the contrary. The following result contrasts with the case of deterministic circuits, where the Distinguishing Paths algorithm uses value injection queries to learn arbitrary transitively reduced acyclic deterministic circuits of constant fan-in over polynomial size alphabets in polynomial time (Angluin et al., 2008b).

Theorem 19 *If* **BPP** $\neq$ **NP** *and* $k \geq 4$ *then there is no polynomial time algorithm using value injection queries that approximately learns all acyclic transitively reduced Boolean probabilistic circuits with fan-in bounded by* k.

Proof Suppose L is a polynomial time algorithm that approximately learns the behavior of every transitively reduced acyclic Boolean probabilistic circuit of fan-in bounded by 4 using value injection queries. The hard computational problem we consider is the following: given a satisfiable 3-CNF formula ϕ over the variables $x_1,\ldots,x_n$ with clauses $c_1,\ldots,c_m$, find an assignment to the variables that satisfies significantly more than seven-eights of the clauses of the formula. Finding such an assignment is **NP**-hard by a result of Håstad (2001). We show how to transform the 3-CNF formula ϕ into a pair of transitively reduced circuits C_0 and C_1 with maximum fan-in 4 such that value injection experiments show a difference that is exponentially small in the depth of the circuits unless we can find a variable assignment that satisfies significantly more than seven-eighths of the clauses of the formula.

The efficiency of our construction depends on the existence of a family of graphs with an expansion property. Specifically, there exists a constant $\alpha < 1$ such that for sufficiently large m, there exists a directed graph G_m on m nodes with constant out-degree 3 such that the second largest eigenvalue λ_2 of the transition matrix for a random walk on G_m satisfies $\lambda_2 \leq \alpha$. Such a family can be constructed by the probabilistic method and explicit constructions are also known; these are surveyed by Hoory, Linial, and Wigderson (2006). Let r be the smallest integer such that $\alpha^r \leq 1/40$.

Let ℓ be a positive integer. The two circuits C_0 and C_1 differ only in their default assignments to a subset of their wires, so we describe their common structure as follows. The circuit consists of a stack of ℓ repetitions of a block consisting of r expander layers above one gadget layer for a total depth of $(2r+1)\ell$. Figure 7 illustrates a block consisting of one expander layer ($r=1$) above a gadget layer. Recall that $x_1,\ldots,x_n$ are the variables of ϕ and $c_1,\ldots,c_m$ are the clauses of ϕ.

A **gadget layer** has three types of wires: inputs $\mathrm{gIn}_1,\ldots,\mathrm{gIn}_m$, variables $\mathrm{x}_1,\ldots,\mathrm{x}_n$, and outputs $\mathrm{gOut}_1,\ldots,\mathrm{gOut}_m$. The input wire gIn_j of each gadget layer except the initial one is identified with

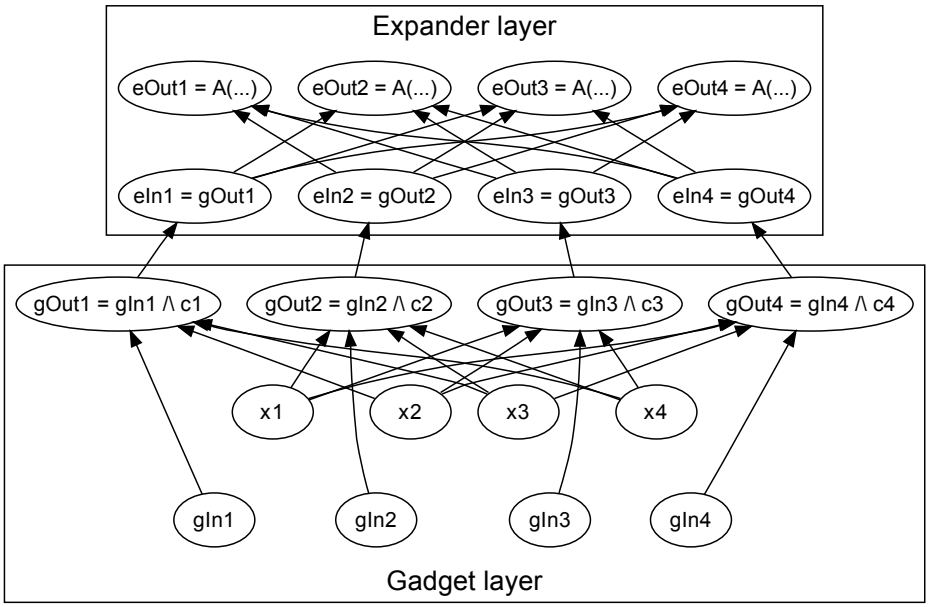


Figure 7: A block with $r = 1$ for the Boolean formula $c_1 \wedge c_2 \wedge c_3 \wedge c_4$, where $c_1 = x_2 \vee \overline{x_3} \vee x_4$ and $c_2 = \overline{x_1} \vee x_3 \vee \overline{x_4}$ and $c_3 = x_1 \vee \overline{x_2} \vee x_4$ and $c_4 = \overline{x_1} \vee x_2 \vee \overline{x_3}$.

the corresponding output wire eOut_j of the expander layer just below it. The variable wires x_i of each gadget layer have no inputs and default to the constant 0. Each output wire gOut_j has four inputs: the corresponding gadget input wire gIn_j and the three variable wires for the variables of the clause c_j of ϕ. Its gate function computes the conjunction of gIn_j and the value of the clause c_j given its three variable values.

Thus, if the learner sets the variable wires x_i in a gadget layer according to a satisfying assignment of ϕ, the signals propagate from the gadget inputs gIn_j to their corresponding outputs gOut_j with perfect fidelity. Otherwise, any unsatisfied clause blocks the signal for the corresponding output.

An **expander layer** averages the outputs of the layer below to be the inputs for the layer above, according to the expander graph G_m. Each input eIn_j of an expander layer is set equal to the corresponding output of the gadget or expander layer immediately below it. The three inputs to eOut_j are eIn_k for the three out-neighbors k of j in the expander graph G_m. The gate function for each eOut_j is the three-input averaging gate $A(x,y,z)$, which is 1 with probability $(x+y+z)/3$. The output of the whole circuit is the first output wire of the final (topmost) expander layer.

The **initial inputs** are the input wires gIn_j of the initial gadget layer. They have no inputs; for the circuit C_0 they are all assigned the default value 0, and for the circuit C_1 they are all assigned the default value 1. Note that C_0 and C_1 are transitively reduced and have a maximum fan-in of 4.

The challenge for the learner is to determine which of C_0 and C_1 is the target circuit. If a value injection experiment succeeds in setting the variable wires in every gadget layer to (possibly different) satisfying assignments for the formula ϕ and leaves all other wires free, then the output of C_0 is 0 and the output of C_1 is 1. If not all the clauses of ϕ are satisfied, then this distance is reduced.

Intuitively, the learner's strategy must be to fix the variable wires in each gadget layer to prevent the signal from the initial inputs from getting blocked; fixing the input or output wires of gadget or expander layers would not help, because they would then have the same value regardless of their inputs. Without a good variable assignment, however, the signal strength drops by a constant factor for each layer, as we now show.

Let e be a value injection experiment. The experiment e induces an assignment to the variables of ϕ for each gadget layer, either by fixing the value of each variable wire or letting it default to 0. The effect of an averaging gate is to select one of its inputs at random and copy the value of that input to the output. Thus, the output of the circuit for experiment e is in effect determined by a random walk backward from the output wire until the walk reaches a wire whose value is fixed by e (and the output is the fixed value), or a gadget layer output wire corresponding to an unsatisfied clause (and the output is 0), or an initial input wire (and the output is the value of that wire.) Suppose that for each gadget layer e encodes a variable assignment that satisfies at most $(9/10)m$ of the clauses of ϕ. We show that the probability that the random walk hits an initial input wire is bounded above by $(39/40)^\ell\sqrt{m}$.

Without loss of generality we may assume that e fixes no wires other than variable wires and initial input wires, because any other fixed wires reduce the probability of reaching an initial input. For $i = 1,\ldots,\ell$, let W_i be the $m \times m$ diagonal matrix with 1s for each satisfied clause in the ith gadget layer and 0s for each unsatisfied clause. Let B be the transition matrix for an r-step random walk on G_m and let $e_1 = (1,0,\ldots,0)$. The probabilities of the random walk hitting the initial inputs are given by the vector $e_1BW_\ell BW_{\ell-1}\cdot BW_2BW_1$. By the following argument, for all i and vectors v, we have $\|vBW_i\| \le (39/40)\|v\|$.

Write $v = cu + w$, where c is a scalar and $u = (1,\ldots,1)$ and w is a vector such that $u \perp w$. Then u is an eigenvector of B with eigenvalue 1 and multiplying w by B shrinks its length to at most the second eigenvalue of B times its original length. By Pythagoras, $\|cu\| \le \|v\|$ and $\|w\| \le \|v\|$. We have $vBW_i = (cu+w)BW_i$. On one hand, $\|cuBW_i\| = \|cuW_i\| \le \sqrt{9/10}\|cu\| \le (19/20)\|v\|$. On the other hand, $\|wB\| \le (1/40)\|w\| \le (1/40)\|v\|$, because the second eigenvalue of B is no larger than $1/40$, and $\|wBW_i\| \le (1/40)\|v\|$, because W_i does not increase the L_2 norm. The resulting $(39/40)^\ell$ bound on the L_2 norm of the probability vector gives a bound of $(39/40)^\ell\sqrt{m}$ on the L_1 norm, which is an upper bound on the probability that any initial input is reached.

Suppose the learning algorithm L runs in time $f(N,1/\varepsilon,1/\delta)$, for some nondecreasing polynomial f, where N is the number of wires in the target circuit. Let $N(\ell)$ denote the number of wires in C_0 (or C_1) as a function of the number ℓ of blocks in the stack. Then $N(\ell) = O(\ell(n+rm))$. We choose ℓ sufficiently large that

$$((39/40)^\ell\sqrt{m})f(N(\ell),4,4) < 1/4,$$

clearly $N(\ell)$ is bounded by a polynomial in m and n.

We randomly and equiprobably choose the target circuit C to be C_0 or C_1 and simulate L with target circuit C and $\varepsilon = \delta = 1/4$. When L makes a value injection experiment e, we check whether any of the induced variable assignments of e satisfies more than $(9/10)m$ clauses of ϕ. If so, we output the assignment and halt. Otherwise, we use a random walk from the output wire in the circuit C to give an output for e. If no experiment e satisfies more than $(9/10)m$ of the clauses of ϕ, then the probability that any of them reaches an initial input in C is less than $1/4$. If none of them reaches an initial input, then L cannot distinguish between C_0 and C_1, and must output a circuit that is not $1/4$-approximately behaviorally equivalent to C with probability at least $3/8 > 1/4$, violating the

requirements of approximate learning. ■

We conclude that if **BPP** $\neq$ **NP**, any polynomial time learning algorithm requires in expectation exponentially many queries in ℓ to learn the default settings of the initial inputs and therefore, PCB is within a polynomial of optimal.

8. Non-Boolean Circuits Revisited

The sharp contrast in results for transitively reduced circuits with alphabet size at least three, for which test paths may show no difference (Lemma 8) and those with alphabet size two, for which test paths must show a significant difference (Lemma 10) motivate us to consider a generalization of the kinds of experiments we consider, to function injection experiments. This generalization allows us to extend the results of Lemma 10 to non-Boolean alphabets.

In a value injection experiment, each wire is either fixed to a constant value or left free. In a function injection experiment for a wire w, these possibilities are expanded to permit a transformation of the value that the wire w would take if it were left free. As an example, consider a transformation in which the values that w could attain are linearly ordered and all values below a certain threshold are mapped to the minimum value and all other values are mapped to the maximum value. It is conceivable that this kind of transformation could be feasible in some domains; in any case, the theoretical consequences are quite interesting. We first give a general definition of function injection, but in the results below we are primarily concerned with 2-partitions, that is, transformations that are like the above example in that they partition the values into at most two blocks and map each block to a fixed element of the block.

An **alphabet transformation** is a function f that maps symbols to distributions over symbols. An alphabet transformation is **deterministic** if it assigns only deterministic distributions, in which case we think of it as a map from symbols to symbols. A deterministic alphabet transformation f is a k-**partition** if there exists a partition of Σ into at most k disjoint nonempty sets Σ_i such that for each i there exists $\sigma_i \in \Sigma_i$ such that $f(\Sigma_i) = \{\sigma_i\}$. Note that if $k_1 \leq k_2$, every k_1-partition is also a k_2-partition.

A 1-partition is a constant function, achieving the same result as fixing the wire to a value in a value injection experiment. We use 2-partitions to reduce the case of larger alphabets to the binary case. Note that the 2-partitions of a binary alphabet include the identity and the two constant functions, but not the negation function.

If D is a value distribution and f is an alphabet transformation, then $f(D)$ is the value distribution in which

$$(f(D))(\sigma) = \sum_{\tau \in \Sigma} D(\tau)(f(\tau))(\sigma).$$

A **function injection experiment** is a mapping e with domain W that assigns to each wire the symbol $*$ or a symbol from Σ or an alphabet transformation f. Then e leaves w **free** if $e(w) = *$, **fixes** w if $e(w) \in \Sigma$, and **transforms** w if $e(w)$ is an alphabet transformation f. We extend the ordering $\leq$ on experiments by stipulating that each alphabet transformation $f \leq *$. A **2-partition experiment** is a function injection experiment in which every alphabet transformation is a 2-partition.

We now define the joint probability distribution on assignments of symbols from Σ to wires determined by a function injection experiment e. If e fixes w, then w is just assigned $e(w)$. Otherwise,

if the inputs of w have been assigned the values $\sigma_1, \ldots, \sigma_k$ and f is the gate function for w, we randomly and independently choose a symbol σ according to the value distribution $f(\sigma_1, \ldots, \sigma_k)$. If w is free in e, then σ is the symbol assigned to w; however, if $e(w)$ is an alphabet transformation, then a symbol τ is chosen randomly and independently according to the value distribution $e(\sigma)$ and assigned to w. That is, when $e(w)$ is an alphabet transformation, we generate the symbol for w as though it were free, and then use the distribution $e(w)$ to transform that symbol. Because C is acyclic, this process assigns a symbol to every wire of C.

In a **function injection query** (FIQ), the learning algorithm gives a function injection experiment e and receives a symbol σ assigned to the output wire of C by the probability distribution defined above. A **functional test path** for a wire w is a function injection experiment in which the free and transformed wires are a directed path in the circuit graph from w to the output wire, and all other wires are fixed.

As an example of how functional test paths help in learning non-Boolean probabilistic circuits, consider again the circuit in the proof of Lemma 8, depicted in Figure 3. We specify a functional test path p by $p(w_1) = p(w_4) = p(w_5) = *$, $p(w_3) = 0$ and $p(w_2)$ is the alphabet transformation $0 \to 0$, $1 \to 0$, and $2 \to 2$. Note that the alphabet transformation is a 2-partition. Then $C(p|_{w_1=0}) = 0$ but $C(p|_{w_1=2}) = U(\{0,1\})$, so this functional test path witnesses a difference of $1/2$, as large as the experiment that leaves all the wires free. Test paths with functions allow us to carry over the results of Lemma 10 to non-Boolean alphabets.

Lemma 20 *Let C be a probabilistic circuit, e be a function injection experiment, w be a wire free in e and D_1, D_2 be value distributions. If there exists $\varepsilon \geq 0$ such that for all functional w-test paths $p \leq e$ that are 2-partitions,*

$$d(C(p|_{w=D_1}), C(p|_{w=D_2})) \leq \varepsilon,$$

then

$$d(C(e|_{w=D_1}), C(e|_{w=D_2})) \leq \kappa(e,w) \cdot \varepsilon.$$

Proof The obstacle in Lemma 10 is that when the alphabet is non-Boolean, we may assume only that D_1 and D_2 have disjoint support, not that they are deterministic. This obstacle can be overcome by injecting a 2-partition at w. Let $\Sigma_1 = \text{support}(D_1)$ and $\Sigma_2 = \text{support}(D_2)$ and assume $\Sigma_1 \cap \Sigma_2 = \emptyset$. Then

$$d(C(e|_{w=D_1}), C(e|_{w=D_2})) \leq \sum_{\substack{\rho_1 \in \Sigma_1 \\ \rho_2 \in \Sigma_2}} D_1(\rho_1) D_2(\rho_2) d(C(e|_{w=\rho_1}), C(e|_{w=\rho_2}))$$

by the triangle inequality. Let

$$(\sigma, \tau) = \arg\max_{\substack{\rho_1 \in \Sigma_1 \\ \rho_2 \in \Sigma_2}} d(C(e|_{w=\rho_1}), C(e|_{w=\rho_2}))$$

so that

$$d(C(e|_{w=D_1}), C(e|_{w=D_2})) \leq d(D_1, D_2) d(C(e|_{w=\sigma}), C(e|_{w=\tau})).$$

Let f be an alphabet transformation that maps Σ_1 to σ and Σ_2 to τ and all other symbols to either σ or τ. Then f is a 2-partition, and

$$d(C(e|_{w=D_1}), C(e|_{w=D_2})) \leq d(C(e|_{w=f(D_1)}), C(e|_{w=f(D_2)})).$$

Since $f(D_1) = \sigma$ and $f(D_2) = \tau$, the rest of the proof goes through. ■

Corollary 21 *Let C be a transitively reduced probabilistic circuit, e be a function injection experiment, w be a wire, and D_1, D_2 be value distributions. If there exists $\varepsilon \geq 0$ such that for all functional w-test paths $p \leq e$,*

$$d(C(p|_{w=D_1}), C(p|_{w=D_2})) \leq \varepsilon,$$

then

$$d(C(e|_{w=D_1}), C(e|_{w=D_2})) \leq \pi(e,w) \cdot \varepsilon.$$

We expect that a further generalization of the Probabilistic Circuit Builder algorithm to use function injection experiments can learn non-Boolean circuits of logarithmic depth and constant fan in in polynomial time. The universal set would map wires to the set containing all alphabet symbols from Σ and all 2-partitions of Σ, of which there are fewer than $|\Sigma|^2 2^{|\Sigma|}$. Thus, the universal set will still be of size $n^{O(1)}$, suggesting that a polynomial time algorithm may be attainable in this case.

Certain other natural questions arise in response to the idea of function injection experiments. We can define circuits C and C' to be **strongly behaviorally equivalent** if $C(e) = C'(e)$ for every function injection query e. Does behavioral equivalence imply strong behavioral equivalence? Once again, alphabet size determines the answer: no for alphabet size greater than two, yes for alphabet size two.

Lemma 22 *For $\Sigma = \{0,1,2\}$, there exist deterministic circuits C_1 and C_2 that are behaviorally equivalent but not strongly behaviorally equivalent.*

Proof In both C_1 and C_2 there are two wires w_1 and w_2, where w_2 is the output wire. In both circuits the gate for w_2 has input w_1 and deterministically maps 0 to 0 and maps 1 and 2 to 1. In C_1, w_1 is the constant 1 and C_2 it is the constant 2.

Then if e is the value injection experiment that leaves both wires free, $C_1(e) = 1 = C_2(e)$. If e fixes either w_1 or w_2, then also $C_1(e) = C_2(e)$. Thus C_1 is behaviorally equivalent to C_2.

However, the 2-partition function injection experiment e that leaves w_2 free and maps the output of w_1 according to the transformation $0 \to 0$, $1 \to 0$, $2 \to 2$ yields $C_1(e) = 0$ and $C_2(e) = 1$. Thus C_1 is not strongly behaviorally equivalent to C_2. ■

However, 2-partition function experiments suffice to establish strong behavioral equivalence.

Lemma 23 *Let C and C' be probabilistic circuits with the same alphabet Σ, the same set of wires and the same output wire. If $C(e) = C'(e)$ for every 2-partition function experiment e then C and C' are strongly behaviorally equivalent.*

Proof By a generalization of the Probabilistic Circuit Builder algorithm to functional test paths. ■

Because in the Boolean case every 2-partition function injection query is a value injection query, we then have the following.

Corollary 24 *For Boolean probabilistic circuits C and C', if C is behaviorally equivalent to C' then C is strongly behaviorally equivalent to C'.*

9. Discussion and Open Problems

These results concern general probabilistic acyclic gates, with no restriction other than fan-in on the kinds of probabilistic gate functions considered. Particular domains may warrant specific assumptions about the gate functions, which may make the learning problems more tractable. For example, for the problem of learning the structure of an independent cascade social network using exact value injection queries, a query-optimal algorithm is presented by Angluin et al. (2008c). Note that social networks may in general contain cycles, which complicates their analysis.

The Distinguishing Paths algorithm (Angluin et al., 2008b) learns transitively reduced acyclic deterministic circuits over polynomial size alphabets with constant fan-in and no depth bound using value injection queries in polynomial time, and relies on a version of the test path lemma. Theorem 19 shows that if **BPP** $\neq$ **NP** then this algorithm does not generalize to arbitrary transitively reduced Boolean probabilistic circuits, but there is a possibility that it might generalize to transitively reduced Boolean probabilistic circuits with a polynomial bound on the total number of directed paths in the circuit graph. A somewhat technical open question is whether in the case of general Boolean probabilistic circuits, the ability to inject the NOT function might reduce the maximum path attenuation to just the number of paths, as it does in the case of the circuit in Figure 6.

Acknowledgments

James Aspnes acknowledges the support of NSF grant CNS-0435201. This work was done while Jiang Chen was a member of the Center for Computational Learning Systems, Columbia University; he acknowledges the support of a research contract from Consolidated Edison. Lev Reyzin acknowledges that this material is based upon work supported under a National Science Foundation Graduate Research Fellowship. A preliminary version of this paper was presented at COLT 2008 (Angluin et al., 2008a). The authors thank the reviewers of the COLT 2008 version of the paper and the referees of the present paper for their thoughtful comments.

References

Tatsuya Akutsu, Satoru Kuhara, Osamu Maruyama, and Satoru Miyano. Identification of genetic networks by strategic gene disruptions and gene overexpressions under a boolean model. *Theor. Comput. Sci.*, 298(1):235–251, 2003.

Dana Angluin and Michael Kharitonov. When won't membership queries help? *J. Comput. Syst. Sci.*, 50(2):336–355, 1995.

Dana Angluin, James Aspnes, Jiang Chen, David Eisenstat, and Lev Reyzin. Learning acyclic probabilistic circuits using test paths. In *Twenty-First Annual Conference on Learning Theory*, pages 169–179. Omicron, July 2008a.

Dana Angluin, James Aspnes, Jiang Chen, and Lev Reyzin. Learning large-alphabet and analog circuits with value injection queries. *Machine Learning*, 72(1-2):113–138, 2008b.

Dana Angluin, James Aspnes, and Lev Reyzin. Optimally learning social networks with activations and suppressions. In *Nineteenth International Conference on Algorithmic Learning Theory*, volume 5254 of *Lecture Notes in Computer Science*, pages 272–286, October 2008c.

Dana Angluin, James Aspnes, Jiang Chen, and Yinghua Wu. Learning a circuit by injecting values. *J. Comput. Syst. Sci.*, 75(1):60–77, 2009.

Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe´er. Using Bayesian networks to analyze expression data. *J. Comput. Biol.*, 7(3–4):601–620, 2000.

Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc. (N.S.)*, 43(4):439–561 (electronic), 2006.

Trey E. Ideker, Vesteinn Thorsson, and Richard M. Karp. Discovery of regulatory interactions through perturbation: Inference and experimental design. In *Pacific Symposium on Biocomputing 5*, pages 302–313, 2000.

David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD '03: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, New York, NY, USA, 2003. ACM.

David Kempe, Jon M. Kleinberg, and Éva Tardos. Influential nodes in a diffusion model for social networks. In *ICALP*, pages 1127–1138, 2005.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

Journal of Machine Learning Research 10 (2009) 1913-1936 Submitted 10/08; Revised 3/09; Published 8/09

Learning Approximate Sequential Patterns for Classification

Zeeshan Syed ZHS@EECS.UMICH.EDU
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109-2121, USA

Piotr Indyk INDYK@CSAIL.MIT.EDU
John Guttag GUTTAG@CSAIL.MIT.EDU
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139-4307, USA

Editor: Charles Elkan

Abstract

In this paper, we present an automated approach to discover patterns that can distinguish between sequences belonging to different labeled groups. Our method searches for approximately conserved motifs that occur with varying statistical properties in positive and negative training examples. We propose a two-step process to discover such patterns. Using locality sensitive hashing (LSH), we first estimate the frequency of all subsequences and their approximate matches within a given Hamming radius in labeled examples. The discriminative ability of each pattern is then assessed from the estimated frequencies by concordance and rank sum testing. The use of LSH to identify approximate matches for each candidate pattern helps reduce the runtime of our method. Space requirements are reduced by decomposing the search problem into an iterative method that uses a single LSH table in memory. We propose two further optimizations to the search for discriminative patterns. Clustering with redundancy based on a 2-approximate solution of the k-center problem decreases the number of overlapping approximate groups while providing exhaustive coverage of the search space. Sequential statistical methods allow the search process to use data from only as many training examples as are needed to assess significance. We evaluated our algorithm on data sets from different applications to discover sequential patterns for classification. On nucleotide sequences from the Drosophila genome compared with random background sequences, our method was able to discover approximate binding sites that were preserved upstream of genes. We observed a similar result in experiments on ChIP-on-chip data. For cardiovascular data from patients admitted with acute coronary syndromes, our pattern discovery approach identified approximately conserved sequences of morphology variations that were predictive of future death in a test population. Our data showed that the use of LSH, clustering, and sequential statistics improved the running time of the search algorithm by an order of magnitude without any noticeable effect on accuracy. These results suggest that our methods may allow for an unsupervised approach to efficiently learn interesting dissimilarities between positive and negative examples that may have a functional role.

Keywords: pattern discovery, motif discovery, locality sensitive hashing, classification

1. Introduction

Pattern discovery has been studied extensively in the context of data mining and knowledge discovery (Han and Kamber, 2005) and causal inference in statistics (Pearl, 2000). The search for patterns is typically guided by classification. The focus is on discovering activity that can distinguish members of a family from non-members (Duda et al., 2000) by identifying activity that is unlikely to occur purely by chance and may have a functional role (Syed et al., 2007).

Pattern discovery has been applied to data from a variety of applications, for example, world wide web transactions (Mobasher et al., 1996), marketing information (Shaw et al., 2001), and medical signals (Li et al., 2005). More recently, there has been an increased interest in applying techniques for discovering patterns to sequences corresponding to genomic data (Wang et al., 1999). Of particular importance in computational biology is the problem of discovering subsequences, that is, motifs, which regulate important biological processes (Kellis et al., 2004). Pattern discovery has been proposed in this context as a machine learning problem (Brazma et al., 1998):

> Given two sets of sequences S^+ and S^- drawn randomly from families F^+ and F^- respectively such that $F^+ \cap F^- = \oslash$, find the pattern W of length L that has high likelihood in F^+ but not in F^-.

This formulation is sufficiently general to apply to a wide variety of applications where sequential data exists. Furthermore, an extensive literature on symbolization (Daw et al., 2003) allows for a large set of time-series signals to be abstracted into sequential data for analysis. We make the notion of a pattern more explicit by refining the goal of pattern discovery described above as follows:

> Given two sets of sequences S^+ and S^- drawn randomly from families F^+ and F^- respectively such that $F^+ \cap F^- = \oslash$, find the subsequence W of length L that occurs with a Hamming distance of at most d with high likelihood in F^+ but not in F^-.

In this paper, we propose a method to efficiently carry out the search for such approximate patterns. A variety of techniques have been proposed to address this problem statement (Lawrence et al., 1993; Bailey and Elkan, 1994; Grundy et al., 1997; Tavazoie et al., 1999; Liu et al., 2001; Pavesi et al., 2001; Sinha and Tompa, 2003). The common strategy adopted by these methods is to approach the problem of pattern discovery by finding activity that is statistically unlikely but occurs consistently in positive examples. Negative examples are primarily used for evaluation. This process means that discriminative patterns in negatively labeled sequences are not explored for classification. Other algorithms for pattern discovery (Delcher et al., 1999; Batzoglou et al., 2000) enumerate all exact patterns across both positive and negative examples to identify sequences that can discriminate between these two cases, but become computationally intractable when allowing subsequences to have an approximate form.

We describe a locality sensitive hashing (LSH) based algorithm to efficiently estimate the frequencies of all approximately conserved subsequences with a certain Hamming radius in both positive and negative examples. The search process attempts to identify patterns that allow maximum discrimination between the two groups. In this way, our method unifies the broad areas of existing work in sequential pattern detection for classification by proposing a way to discover patterns that are both approximate and derived using the additional information available in negative instances.

LSH forms a key component of our method. The use of LSH has been proposed earlier in the context of pattern discovery to identify interesting activity in positive examples (Buhler, 2001; Buhler and Tompa, 2002). We supplement this work by allowing for information from negative examples to be factored into the search and by proposing different optimizations to the search process. In particular, we expand the use of LSH in pattern discovery from indexing to fast counting and approximate clustering. While LSH provides runtime efficiency to the search process, it imposes significant space requirements, and we describe an iterative method that uses a single LSH table in memory to address this issue. We also explore the idea of using clustering as part of pattern discovery to reduce approximate subsequences with significantly overlapping Hamming radii to a small number. This aspect of our work resembles efforts for web clustering (Haveliwala et al., 2000). We explore similar ideas within the context of approximate pattern discovery. This decreases the number of motifs to be evaluated while still providing a fairly exhaustive coverage of the search space. We describe a clustering method based on a 2-approximate solution of the k-center problem to achieve this goal.

In addition to LSH and clustering, we also draw upon sequential statistical methods to make the search for interesting patterns more efficient. The process of identifying patterns with discriminative ability makes use of concordance and rank sum testing. In many cases, the goodness of approximate patterns can be assessed without using data from all training sequences. We propose a further optimization to address these cases. The runtime and space requirements of the pattern discovery process can be reduced by using sequential statistical methods that allow the search process for patterns to terminate after using data from only as many training examples as are needed to assess significance.

We address a similar goal to earlier work on using hypergeometric significance testing to discover patterns that are enriched in a positive set relative to a negative set (Barash et al., 2001). The focus of this work is to generate seeds of short lengths that can be expanded using an expectation-maximization (EM)-like process to produce a position specific scoring matrix of the desired length. However, in contrast to our work, this method is based on the assumption that a pattern occurs at most once in each sequence. This leads it to disregard multiple copies of a match within the same sequence. Moreover, the use of a testing function based on hypergeometric analysis may affect the accuracy of this method (Leung and Chin, 2006).

Our algorithm to find approximate discriminative patterns is also related to previous work on the use of profile hidden Markov models (Krogh, 1994; Jaakkola et al., 1999) to optimize recognition of positively and negatively labeled sequences. This work focuses on learning the parameters of a hidden Markov model that can represent approximations of subsequences. Generally, this approach requires large amounts of data or sophisticated priors to train the hidden Markov model. Computing forward and backward probabilities from the Baum-Welch algorithm is also very computationally intensive. Subsequent work in this area focuses on mismatch tree-based kernels (Leslie et al., 2003) for use in a support vector machine (SVM) classifier. This work focuses on efficiently calculating a kernel based on the mismatch tree data structure (Eskin and Pevzner, 2002), which quantifies how different two sequences are based on the approximate occurrence of the fixed L-length subsequences within them. The mismatch kernel is used to train an SVM and assign labels to unknown query sequences.

Our algorithm supplements this work by measuring how frequently each subsequence occurs in an approximate form in the data. In contrast to the mismatch kernel, which focuses on quantifying the difference between two sequences and does not report the frequency of individual approximate

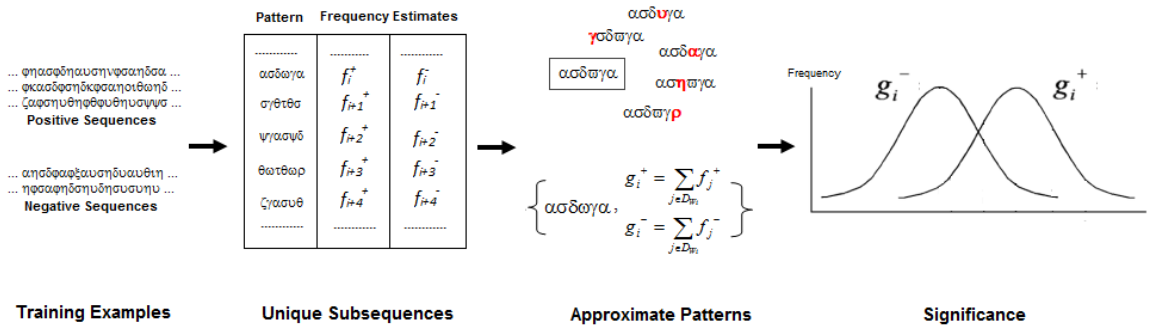


Figure 1: Overview of the pattern discovery process.

subsequences in the data, our algorithm focuses on identifying the specific approximate patterns with discriminative value. This approach can be integrated more easily with the use of sequential statistics, that is, since the frequencies of each approximate pattern are retained during analysis, this information can be used to determine if patterns are good or bad discriminators without analyzing all the available data.

We evaluated our algorithm on data sets from different applications to discover sequential patterns for classification. On nucleotide sequences from the Drosophila genome, our method was able to discover binding sites for genes that are preserved across the genome and do not occur in random background sequences. On symbolized electrocardiographic time-series from patients with acute coronary syndromes, our pattern discovery approach identified approximately conserved sequences of morphology variations that were predictive of future death in a test population. These results suggest that our methods may allow for an unsupervised approach to learn interesting dissimilarities between positive and negative examples that may have a functional role.

The rest of this paper is organized as follows: Section 2 gives an overview of our algorithm. Section 3 describes a locality sensitive hashing scheme to find approximate matches to all subsequences in the data set. Section 4 proposes the use of clustering to reduce the number of approximate patterns analyzed during pattern discovery. Section 5 discusses the statistical approaches used in assessing the goodness of patterns. Section 6 details the evaluation methodology of our pattern discovery algorithm on data from different real-world applications. Section 7 reports the results of this study. Section 8 concludes with a discussion.

2. Overview

The process of discovering discriminative patterns of a specified length L from positive and negative sequences is carried out in two stages: frequency estimation and pattern ranking. Figure 1 presents an overview of this approach.

2.1 Frequency Estimation

Given a set of positive examples $S^+ = \{S_x^+ | x = 1, \ldots, N^+\}$ and a set of negative examples $S^- = \{S_y^- | y = 1, \ldots, N^-\}$ the frequency estimation step measures the frequency of every unique subse-

quence W_i for $i = 1, \ldots, M$ in sequences belonging to S^+ and S^-. The resulting frequency for W_i in positive and negative examples is denoted as:

$$f_i^+ = \{f_{i,z}^+ | z \in S^+\},$$
$$f_i^- = \{f_{i,z}^- | z \in S^-\}$$

where $f_{i,z}^+$ and $f_{i,z}^-$ are the frequencies with which W_i appears in sequences z drawn from S^+ and S^-, and f_i^+ and f_i^- are vectors measuring the frequency of W_i in all positive and negative sequences.

To allow for approximate patterns, unique subsequences are then matched to all other subsequences at a Hamming distance of at most d from W_i. Denoting this group of subsequences as D_{W_i}, the resulting frequency for the subsequence W_i and its approximate matches is defined as:

$$g_i^+ = \sum_{j \in D_{W_i}} f_j^+,$$
$$g_i^- = \sum_{j \in D_{W_i}} f_j^-$$

where g_i^+ and g_i^- are vectors obtained by summing up the vectors f_i^+ and f_i^- for all subsequences within a given Hamming radius d of W_i.

In Section 3, we describe an LSH-based solution that allows for efficient discovery of the subsequences D_{W_i} matching W_i. We also present a clustering approach in Section 4 to reduce overlapping approximate patterns for which frequencies are estimated to a smaller number with less redundancy for subsequent analysis.

2.2 Pattern Ranking

The goal of the search process is to identify approximately matching subsequences that can discriminate between positive and negative training examples. The pattern ranking stage therefore scores each candidate approximate pattern according to its discriminative ability. We use two measures to assess the goodness of patterns.

The first approach to score patterns is to use rank sum testing. This technique is a non-parametric approach for assessing whether two samples of observations come from the same distribution. Patterns are ordered based on the significance of separation (as measured by the p-value) obtained by rank sum testing. A second scoring criterion used by our work is the C-statistic, which corresponds to the area under the receiver operating characteristic (ROC) curve. Details of these techniques are provided in Section 5. We further describe how sequential methods can be used to reduce the search process to only process as many training examples as are needed to determine if a candidate pattern has high or low discriminative ability.

3. Locality Sensitive Hashing

In this section, we describe the use of locality sensitive hashing in our algorithm.

3.1 Finding Approximate Matches for a Subsequence

Locality sensitive hashing (Indyk and Motwani, 1998) has been proposed as a randomized approximation algorithm to solve the nearest neighbor problem. Given a set of subsequences, the goal of

LSH is to pre-process the data so that future queries searching for closest points under some l_p norm can be answered efficiently. A brief review of LSH is presented here.

Given two subsequences S_x and S_y of length L, we describe them as being similar if they have a Hamming distance of at most d. To detect similarity, we choose K indices $i_1, \ldots, i_K$ at random with replacement from the set $\{1, \ldots, L\}$. The locality sensitive hash function $LSH(S)$ is then defined as:

$$LSH(S) = < S[i_1], \ldots, S[i_k] >$$

where $< \ldots >$ corresponds to the concatenation operator. Under this scheme, S_x and S_y are declared to be similar if:

$$LSH(S_x) = LSH(S_y). \tag{1}$$

The equality in Equation 1 corresponds to an exact match. However, since the indices used by the locality sensitive hash function $LSH(S)$ may not span the entire subsequences S_x and S_y, an exact match in Equation 1 may be obtained if S_x and S_y match approximately.

Practically, LSH is implemented by creating a hash table using the $LSH(S)$ values for all subsequences as the keys. Searching for the approximate neighbors of a query subsequence corresponds to a two-step process. The locality sensitive hash function is first applied to the query. Following this, the bucket to which the query is mapped is searched for all original subsequences with a Hamming distance of at most d.

Two subsequences with a Hamming distance of d or less may not match for a random choice of K indices if one of the K indices chosen corresponds to a position in which S_x and S_y differ. The probability of such a miss is bounded by (Indyk and Motwani, 1998):

$$Pr[LSH(S_x) \neq LSH(S_y)] \leq [1 - (1 - \frac{d}{L})^K].$$

By repeating the process of choosing K indices T times this probability can be reduced further to:

$$Pr[LSH(S_x) \neq LSH(S_y)] \leq [1 - (1 - \frac{d}{L})^K]^T. \tag{2}$$

Effectively, Equation 2 corresponds to constructing a data structure comprising T hash tables using different locality sensitive hash functions $LSH_1(S), \ldots, LSH_T(S)$. Approximate neighbors for a query are detected by searching for matches in each of these hash tables as described earlier.

The intuition underlying LSH is that the problem of searching through all possible subsequences in the data set for a match can be reduced to the more feasible problem of first rapidly identifying a small set of potential matches with a bounded error, and then searching through this smaller set to remove false positives. The lower the desired error bound for false negatives affecting correctness (i.e., by choosing K and T), the higher the corresponding false positive rate affecting the runtime of the algorithm. The choice between these two parameters depends on the application and the underlying data set.

3.2 Finding Approximate Matches Between All Subsequences

LSH provides an efficient mechanism to find the nearest neighbors of a given subsequence. To find the nearest neighbors for all M subsequences in the data set, each member of the set can be passed

through the entire LSH data structure comprising T hash tables for matches. Unfortunately, this process is both computationally and memory intensive. In what follows, we describe a strategy to reduce the space requirements of LSH-based search for all approximate matches between subsequences. Section 4 further addresses runtime issues by proposing a clustering amendment to the search process.

Different approaches have been proposed recently to reduce the space requirements of LSH. In particular, the use of multi-probe LSH (Lv et al., 2007) has been shown to substantially reduce the memory requirements for traditional LSH by searching each LSH hash table (i.e., corresponding to a random selection of K indices) more thoroughly for misses. This additional work translates into fewer LSH hash tables being needed to bound the given error rate. As a result, the space of the LSH data structure decreases.

In our work, the memory requirements of LSH are reduced by organizing the approximate matching process as T iterations. Each iteration makes use of a single locality sensitive hash function and maintains only a single hash table in memory at any time. To preserve state across iterations, the search process maintains a list of matching pairs found during each loop after removing false positives. The subsequences D_{W_i} matching W_i are found as:

$$D_{W_i} = \bigcup_{t=1}^{T} \{W_j | LSH_t(W_j) = LSH_t(W_i)\}.$$

4. Clustering Subsequences

The runtime of the pattern discovery process as described so far is dominated by the approximate matching of all subsequences. Every subsequence is first used to create the LSH data structure, and then passed through the LSH data structure to find matches with a Hamming distance of at most d. This process is associated with considerable redundancy, as matches are sought individually for subsequences that are similar to each other. The overlap between approximate patterns increases the computational needs of the pattern discovery process and also makes it more challenging to interpret the results as good patterns may appear many times in the output.

To address this issue, we reduce patterns to a much smaller group that still collectively spans the search space. This is done by making use of a clustering method based on a 2-approximate solution to the k-center problem. The focus of this clustering is to group together the original subsequences falling into the same hash bucket during the first LSH iteration. Each of the clusters obtained at the end of this process corresponds to an approximate pattern that is retained. During subsequent iterations, while all subsequences are still used to construct the LSH tables, only the cluster centroids are passed through the LSH data structure. This reduces the runtime of the search by reducing the number of times subsequences have to be passed through the LSH tables to find true and false positives. It also reduces the memory requirements of the search by reducing the number of subsequences for which we need to maintain state about approximate matches.

The traditional k-center problem can be formally posed as follows. Given a complete graph $G = (V,E)$ with edge weights $\omega_e \geq 0$, $e \in E$ and $\omega(v,v) = 0$, $v \in V$, the k-center problem is to find a subset $Z \in V$ of size at most k such that the following quantity is minimized:

$$W(Z) = \max_{i \in V} \min_{j \in Z} \omega_{(i,j)}.. \tag{3}$$

The k-center problem is NP-hard, but a 2-approximate solution has been proposed (Hochbaum and Shmoys, 1985) for the case where the triangular inequality holds:

$$\omega_{(i,j)} + \omega_{(j,k)} \geq \omega_{(i,k)}.$$

The Hamming distance metric obeys the triangular inequality. Under this condition, the process of clustering can be decomposed into two stages. During the first LSH iteration, we identify subsequences that serve as cluster seeds using the 2-approximate solution to the k-center problem. Subsequent LSH iterations are used to grow the clusters till the probability that any subsequence within a Hamming distance at most d of the cluster centroid is missed becomes small. This approach can be considered as being identical to choosing a set of subsequences during the first LSH iteration, and finding their approximate matches by multiple LSH iterations.

More formally, during the first LSH iteration, for each bucket b_i in the hash table for $i = 1, \ldots, B$, we solve the k-center problem using the 2-approximate method (Hochbaum and Shmoys, 1985) with a Hamming distance metric. The number of subsequences forming centers k_i for the i-th hash table bucket is determined alongside the specific centroid subsequences from:

$$k_i = \min\{k | W(z_i(k)) \leq d\} \tag{4}$$

where $W(Z)$ is defined as in Equation 3 and $z_{i(k)}$ denotes the subsequence centers chosen for a particular choice of k in Equation 4, that is:

$$k_i = \min\{k | \max_{j \in b_i} \min_{z_i(k)} \omega_{(j,z_i(k))} \leq d\}.$$

The final set of subsequences chosen as centroids at the end of the first LSH iteration then corresponds to:

$$\Phi = \bigcup_{i=1}^{B} z_i(k_i).$$

The LSH iterations that follow find approximate matches to the subsequences in Φ. It is important to note that while clustering reduces a large number of overlapping approximate patterns to a much smaller group, the clusters formed during this process may still overlap. This overlap corresponds to missed approximate matches that do not hash to a single bucket during the first LSH iteration. Techniques to merge clusters can be used at the end of the first LSH iteration to reduce overlap. In our work, we tolerate small amounts of overlap between clusters analogous to the use of sliding windows to more thoroughly span the search space. Figure 2 illustrates the clustering process.

5. Pattern Ranking

Given the frequencies g_i^+ and g_i^- of an approximate pattern corresponding to all subsequences within a Hamming distance d of the subsequence W_i, a score can be assigned to the pattern by using concordance statistics and rank sum testing.

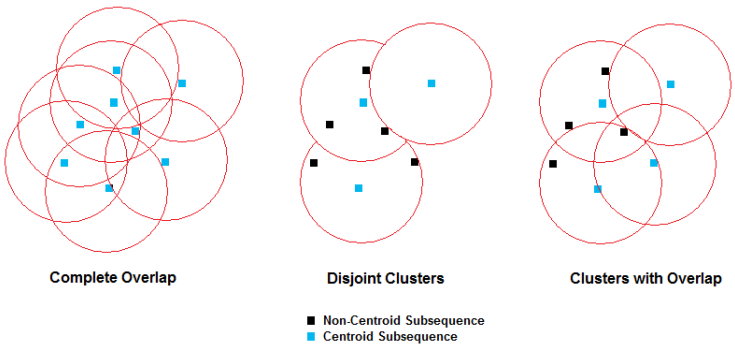


Figure 2: In the absence of clustering there is significant redundancy between the Hamming radii of approximate patterns. Partitioning the data into disjoint clusters can help address this issue. In our work, we reduce the original approximate patterns into a small group with some overlap to span the search space.

5.1 Concordance Statistic

The concordance statistic (C-statistic) (Hanley and McNeil, 1982) measures the discriminative ability of a feature to classify binary endpoints. The C-statistic corresponds to the area under the receiver operating characteristic (ROC) curve, which describes the inherent trade-off between sensitivity and specificity. As opposed to measuring the performance of a particular classifier, the C-statistic directly measures the goodness of a feature (in this case the frequency with which an approximate pattern occurs) by evaluating its average sensitivity over all possible specificities.

The C-statistic ranges from 0-1. A pattern that is randomly associated with the labels would have a C-statistic of 0.5. Conversely, good discriminators would correspond to either low or high C-statistic values.

5.2 Rank Sum Testing

An alternate approach to assess the goodness of patterns is to make use of rank sum testing (Wilcoxon, 1945; Lehmann, 1975). This corresponds to a non-parametric method to test whether a pattern occurs with statistically different frequencies in both positive and negative examples.

Given the frequencies g_i^+ and g_i^- of an approximate pattern in both positive and negative examples, the null and alternate hypotheses correspond to:

$$H_0 : \mu_{g_i^+} = \mu_{g_i^-},$$
$$H_1 : \mu_{g_i^+} \neq \mu_{g_i^-}.$$

Rank sum testing calculates the statistic U whose distribution under H_0 is known. This is done by arranging the g_i^+ and g_i^- into a single ranked series. The ranks for the observations from the g_i^+ series are added up. Denoting this value as R^+ and the number of positive examples by N^+, the statistic U is given by:

$$U = R^+ - \frac{N^+(N^+ + 1)}{2}.$$

The obtained value of U is compared to the known distribution under H_0 and a probability for this observation corresponding to the null hypothesis is obtained (i.e., the p-value). For large samples, U is approximately normally distributed and its standardized value can be checked in tables of the normal distribution (Gibbons, 1985; Hollander and Wolfe, 1999). The lower the p-value obtained through rank sum testing, the more probable it is that g_i^+ and g_i^- have distributions with different means, that is, that the approximate pattern is distributed differently in positive and negative examples.

5.3 Sequential Statistical Tests

The runtime and space requirements of the pattern discovery process can be reduced by analyzing only a subset of the training data. For example, it may be possible to recognize patterns with high discriminative ability without the need to analyze all positive and negative examples.

Our pattern discovery algorithm starts out by using a small subset of the initial training data for batch analysis. This helps identify candidate approximate patterns that occur in the data set and collectively span the search space. The remaining training examples are used for scoring purposes only. These examples are added in an online manner and during each iteration, the occurrence of outstanding candidate patterns in positive and negative examples is updated. Patterns may then be marked as being good or bad (and consequently removed from further analysis) or studied further to resolve uncertainty. The number of candidate patterns is therefore monotonically non-increasing with iteration number. As a result, the analysis of training examples becomes faster as more data is added, since fewer patterns need to be scored.

A sequential formulation for rank sum testing has been proposed (Phatarfod and Sudbury, 1988) that adds positive and negative examples in pairs. The frequencies of an approximate pattern in positive and negative examples at the end of iteration n can be denoted as $g_i^+(j)$ and $g_i^-(j)$ where $j = 1, \ldots, n$. The corresponding statistic for rank sum testing is:

$$U_n = \sum_{x=1}^{n} \sum_{y=1}^{n} I(g_i^+(x) > g_i^-(y)). \tag{5}$$

The operator $I(.)$ is equal to one when the inequality holds and zero otherwise. Using this statistic, the decision at the end of the n-th iteration corresponds to accepting H_0 if:

$$\frac{U_n}{n} < \frac{n}{2} - \lambda \log(\frac{1-\beta}{\alpha}) \tag{6}$$

while H_1 is accepted if:

$$\frac{U_n}{n} > \frac{n}{2} - \lambda \log(\frac{\beta}{1-\alpha})$$

where λ is defined as Phatarfod and Sudbury (1988):

$$\lambda = \frac{1 - \frac{\delta^2}{2} - \frac{\delta^3}{3\sqrt{3}} - \frac{\delta^4}{48}}{2\sqrt{3}\delta - \frac{\delta^2}{2}}. \tag{7}$$

In Equations 6 to 7, α and β correspond to desired false positive and false negative rates for the sequential rank sum testing process, while δ is a user-specified parameter that reflects the preferences of the user regarding how fine or coarse a difference the distributions are allowed to have under the alternate hypothesis. If both inequalities are not met, the process of adding data continues until there are no more training examples to add. In this case, all outstanding candidate patterns are rejected.

This formulation of sequential rank sum testing adds data in pairs of positive and negative examples. In cases where there is a skew in the training examples (without loss of generalization we assume a much larger number of negative examples than positive ones), we use a different formulation of sequential testing (Reynolds, 1975). Denoting the mean frequency in positive training samples as:

$$\mu_i^+ = \sum_{x=1}^{N^+} g_i^+(x).. \tag{8}$$

The alternate hypothesis can be redefined as the case where $h_i = g_i^- - \mu_i^+$ is asymmetrically distributed about the origin. This can be identified using the statistic:

$$U_n = \sum_{x=1}^{n} \frac{1}{x+1} \text{sgn}(h_i(x)) R_{xx} \tag{9}$$

where R_{xy} is defined as the rank of $|h_i(x)|$ in the set $\{|h_i(1)|, \ldots, |h_i(y)|\}$ with $x \leq y$ and $\text{sgn}(|h_i(x)|)$ is 1 if $h_i(x) \geq 0$ and -1 otherwise. The test procedure using the statistic continues taking observations as long as $U_n \in (-\delta, \delta)$ where δ is a user-specified parameter.

Traditional formulations of sequential significance testing remove good patterns from analysis when the test statistic is first found to lie above or below a given threshold. Patterns with potentially low discriminative ability are retained for further analysis and are discarded if they do not meet any of the admission criteria during any iteration of the search process. Since there may be a large number of such patterns with low discriminative value, we make use of a modified approach to reject poor hypotheses while retaining patterns that may have value in classification. This strategy improves efficiency, while also ensuring that good patterns are ranked using the available training data. Given the typical goal of pattern discovery to return the best patterns found during the search process, this technique naturally addresses the problem statement.

Given the test statistics in Equations 5 and 9, we remove all patterns that have a test statistic:

$$U_n < \lambda U_{\max}$$

where $U_{\max}$ is the maximum test statistic for any pattern and λ is a user-specific fraction (e.g., 0.2).

5.4 Multiple Hypothesis Correction

While assessing a large number of approximate patterns, M, the statistical significance required for goodness must be adjusted for Type I (i.e., false positive) errors. If we declare a pattern to be significant for some probability of the null hypothesis less than θ, then the overall false positive rate for the experiment assuming independence of patterns is given by:

$$FP = 1 - (1-\theta)^M.$$

If we do not assume that the patterns are independent, the false positive rate can be bounded by:

$$FP \leq \theta M.$$

To account for this condition, a more restrictive level of significance must be set consistent with the number of unique patterns being evaluated (in our case, the clusters obtained earlier). If c clusters are assessed for goodness, the Bonferroni correction (Bland and Altman, 1995) suggests that the level of significance be set at:

$$\theta' = \frac{\theta}{c}.$$

This addresses the issue of increasing false positives caused by the evaluation of a large number of clusters by correspondingly lowering the p-value required to accept a pattern.

6. Evaluation

All experiments were carried out on a 3.2 GHz P4 with 2 GB of RAM running Linux Fedora Core4. The algorithms to be evaluated were implemented in Java.

6.1 Regulatory Motifs in Drosophila Genome

We evaluated our pattern discovery algorithm on data from the Assessment of Computational Motif Discovery Tools project (Li and Tompa, 2006; Tompa et al., 2005). The focus of this project was to compare 13 motif discovery tools on nucleotide sequences from the human, mouse, fly and yeast genomes to see if they could successfully identify known regulatory elements such as binding sites for genes. These tools differed from each other mainly in their definition of a motif, and in the method used to search for statistically overrepresented motifs. Authors with specific expertise were chosen to test each tool and avoid the disadvantage of being run with an uninformed set of parameters. The expert predictions were then compared with known binding sites in the TRANSFAC database (Wingender et al., 1996) using various statistics to assess the correctness of the predictions.

In our work, we focused on nucleotide sequences from the *Drosophila melanogaster* genome comprising 43 kb base pairs (Li and Tompa, 2006; Tompa et al., 2005). These sequences were divided into 8 data sets, each corresponding to a different binding site. The Drosophila genome was the smallest (in terms of the available data) of the four genomes used in the project, allowing us to compare our method with more computationally intensive algorithms that do not use LSH or clustering. To use our method, we generated random negative sequences with the same background frequencies and lengths as the positive examples, and supplemented both positive and negative sequences with their reverse complements. Our method predicted binding sites in the original data corresponding to all subsequences in the group D_{W_i} with maximum discriminative value. While our algorithm has been designed to use additional information in negative examples when available, this approach presents an example of how our method can be used even in cases where only positive training data is present and negative examples are generated using a simple approach.

The pattern discovery process attempted to find approximate subsequences of lengths 8, 12 and 16. We investigated Hamming radii of 1-3 for patterns of length 8, 2-4 for those of length 12 and 3-5 for length 16. Parameters were chosen using the inequality in Equation 2 so that the LSH probability of false negatives in each case was below 0.005. For each of these cases, we ran our algorithm three

times and consistent with the other motif discovery algorithms evaluated, we chose only the best pattern discovered (where best corresponded to lowest rank sum p-value or no result if a pattern with a p-value less than 0.05 was not found).

We compared our method to the 13 motif discovery tools evaluated on the same data set using the *nPC* and *nCC* summary statistics (Li and Tompa, 2006; Tompa et al., 2005). Given the number of nucleotide positions in both known sites and predicted sites (*nTP*), the number of nucleotide positions in neither known sites nor predicted sites (*nTN*), the number of nucleotide positions not in known sites but in predicted sites (*nFP*) and the number of nucleotide positions in known sites but not in predicted sites (*nFN*), the nucleotide level performance coefficient (*nPC*) (Pevzner and Sze, 2000) was defined as:

$$nPC = \frac{nTP}{nTP + nFN + nFP}.$$

The nucleotide level correlation coefficient (*nCC*) (Burset and Guigo, 1996) was defined as:

$$nCC = \frac{nTP.nTN - nFN.nFP}{\sqrt{(nTP+nFN)(nTN+nFP)(nTP+nFP)(nTN+nFN)}}.$$

In addition to comparing our method with results from the 13 motif discovery algorithms evaluated in the Assessment of Computational Motif Discovery Tools project, we also explored the runtime and accuracy of two variations of our algorithm. These variations were meant to assess the contribution of clustering and LSH-based approximate matching to the performance of our method. The first variation we examined did not make use of the clustering process described in Section 4 to reduce overlapping approximate patterns to a smaller group. The second variation further avoided the use of LSH to match subsequences and was based instead on an exhaustive search.

While comparing the three approaches, we use the following notation: *LSHCS* for our original algorithm using clustering and sequential statistics, *NoClust* for the variation that did not use clustering, and *ExhSearch* for the exhaustive search algorithm that further avoided the use of LSH to match subsequences.

For some binding site data sets *NoClust* and *ExhSearch* did not terminate even after very long processing times. We terminated algorithms if they did not produce results within 24 hours of CPU time. These cases are annotated where included.

6.2 Regulatory Motifs in ChIP-on-chip Yeast Data

We evaluated the ability of our method to discover yeast transcription factor binding motifs in ChIP-on-chip data (Harbison et al., 2004). The "gold standard" motifs for this data set were generated by applying a suite of six motif-finding algorithms to intergenic regions from *Saccharomyces cerevisiae* and clustering the results to arrive at a consensus motif for each transcription factor. When no motif was found computationally for the intergenic regions, a literature-based consensus motif was used.

In our experiments, we focused on the 21 transcription factors in the data set for which the six motif-finding algorithms in Harbison et al. (2004) failed to find a significant motif and the reported motif had to be based on a consensus obtained from the literature. For all tests, we used the output from the *LSHCS* algorithm with the lowest rank sum p-value (or no result if a motif with p-value less than 0.05 was not found), with the motif width chosen to match the width of the literature consensus motif. This approach was analogous to earlier work on ChIP-on-chip data to evaluate motif discovery methods (Siddharthan et al., 2005).

For each experiment, we defined positive examples as the set of probe sequences found to bind the transcription factor and set negative examples to be randomly selected non-binding probe sequences. The positive and negative sets contained the same number of sequences. This approach was also chosen to be consistent with earlier studies to evaluate motif discovery tools (Redhead and Bailey, 2007).

6.3 Predictive Morphology Variations in Electrocardiogram Signals

We evaluated our method on 24-hour electrocardiographic (ECG) signals from patients admitted with non-ST-elevation acute coronary syndromes (NSTEACS) to discover patterns of morphology changes associated with future cardiovascular death. Earlier studies suggest that increased beat-to-beat variations in ECG morphology may be associated with instability in the conduction system of the heart and could help identify high risk patients (Syed et al., 2008). We applied our pattern discovery algorithm to discover specific sequences of beat-to-beat changes that had predictive value.

Given a 24-hour ECG signal, we first converted the data recorded during the course of hospitalization into a time-series of morphology changes between pairs of consecutive heart beats (Syed et al., 2008). Morphology changes between beats were measured using a dynamic time-warping algorithm (Rabiner et al., 1978) that calculates the time-aligned energy difference between beats. The morphology change time-series was then converted into a sequence using symbolic aggregate approximation (SAX) (Lin et al., 2003) with an alphabet size of 10. In this manner, multiple ECG signals were transformed into sequences that could be analyzed by our method. On average, each sequence corresponding to 24 hours of ECG was almost 100,000 symbols long.

On a training set of 765 patients, where 15 patients died over a 90 day period following NSTEACS (i.e., 15 positive examples and 750 negative examples), we used our pattern discovery algorithm to learn sequences of morphology changes in the ECG signal that were predictive of death. We searched for patterns of length 8 with a Hamming distance of at most 2. Parameters were chosen using the inequality in Equation 2 so that the LSH probability of false negatives was less than 0.01. We selected patterns that showed a C-statistic greater than 0.7 and a rank sum test p-value of 0.05 corrected for multiple hypotheses using the Bonferroni correction. These were evaluated on a test set of 250 patients with 10 deaths.

We also studied the runtime performance of our algorithm on this data set with and without the use of sequential statistics to find significant patterns. Given the large number of negative examples in the training data, we employed the sequential formulation in Equations 8 and 9 with $\lambda = 0.2$. Consistent with the notation proposed in Section 6 we denote our original algorithm using sequential statistics as *LSHCS* while the variation that avoids sequential statistics is denoted by *NoSeqStats*.

7. Results

The results of our experiments are as follow.

7.1 Regulatory Motifs in Drosophila Genome

Table 1 presents the results of the 13 different motif discovery algorithms evaluated by the Assessment of Computational Motif Discovery Tools project. The results of using our method are given in Table 2.

Algorithm	*nPC*	*nCC*
AlignACE	0	-0.006
ANN-Spec	0.010	0.002
Consensus	0	-0.011
GLAM	0.002	-0.008
Improbizer	0.008	0.002
MEME	0.021	0.027
MEME3	0.016	0.013
MITRA	0	-0.008
MotifSampler	0.003	-0.006
Oligodyad-Analysis	0	-0.015
QuickScore	0	-0.016
SeSiMCMC	0.036	0.054
Weeder	0.009	0.011
YMF	0	-0.014

Table 1: Performance of 13 different motif discovery methods on the Drosophila genome (*nPC*=performance coefficient, *nCC*=correlation coefficient) in the Assessment of Computational Motif Discovery Tools project (Tompa et al., 2005).

Parameter	*nPC*	*nCC*
$L = 8, d = 1$	0.021	0.031
$L = 8, d = 2$	0.007	-0.009
$L = 8, d = 3$	0.013	-0.002
$L = 12, d = 2$	0.013	0.018
$L = 12, d = 3$	0.039	0.062
$L = 12, d = 4$	0.013	0.018
$L = 16, d = 3$	0	-0.011
$L = 16, d = 4$	0.032	0.055
$L = 16, d = 5$	0.056	0.093

Table 2: Performance of the *LSHCS* pattern discovery method on the Drosophila genome using different input parameters (*nPC*=performance coefficient, *nCC*=correlation coefficient, L=pattern length, d=maximum Hamming distance allowed in pattern).

Our *LSHCS* algorithm compared favorably with the other motif discovery algorithms evaluated on the same data by experts. In particular, for two of the parameter settings evaluated (i.e., $L = 12, d = 3$ and $L = 16, d = 5$), our algorithm had both a higher performance coefficient and a higher correlation coefficient than any of the other methods. The $L = 16, d = 4$ case also had a higher correlation coefficient than the motif discovery algorithms previously reported, although the performance coefficient for this choice of parameters was exceeded by *SeSiMCMC*.

We note that these findings help only to validate the ability of our pattern discovery approach to identify potentially interesting activity. Since these results hold on a specific genome, we caution against interpreting the results as a more general comparison of *LSHCS* with nucleotide motif discovery methods. In particular, we observe that the nucleotide motif discovery methods in Table 1 were tested blindly on a single pre-determined choice of parameters. It is possible that with different choices of input parameters (i.e., similar to the evaluation of *LSHCS* using different values of L and d), these methods would have yielded significantly better results.

Tables 3 and 4 present the performance and correlation coefficients for the *NoClust* and *ExhSearch* algorithms. For both these variations, the algorithms did not terminate for the largest choice of Hamming radius d. Investigation revealed that the slow processing times in these cases were associated with insufficient memory to store the neighborhoods for all approximate patterns (i.e., in the absence of clustering for both algorithms). For large choices of the maximum allowed Hamming radius, the neighborhood for each pattern is more extensive and storing this information for all overlapping patterns imposes significant space requirements. Conversely, by using clustering, *LSHCS* is able to reduce the number of overlapping patterns and avoid references to disk.

We found some of the results in Tables 3 and 4 comparing *LSHCS* to *NoClust* and *ExhSearch* to be surprising. In contrast to *LSHCS*, the *NoClust* variation explores all overlapping patterns while *ExhSearch* uses an exhaustive search to find nearest neighbors (i.e., avoiding the small false negative probability associated with LSH). Since both variations use strictly more data and explore the outputs of *LSHCS* as well as other alternatives, we expected the results to improve uniformly between *LSHCS* and *ExhSearch* for all parameter selections. While this was generally the case, for some parameter choices (e.g., $L = 16, d = 4$) the best results were obtained by *LSHCS*. Examining the outputs by all three algorithms revealed these inconsistencies to be the result of resolving ties between patterns with identical rank sum p-values arbitrarily. While presenting the results in this section, we explicitly note this limitation of the objective function (i.e., the rank sum p-value) used for evaluation.

Ignoring parameter choices for which *NoClust* and *ExhSearch* did not terminate, the performance of all three methods was similar as shown in Table 5. A comparison of the running time for all three algorithms is also presented in Table 6. The running time increased significantly both as all overlapping approximate patterns were studied, and when an exhaustive search was used to replace LSH. The increase in runtime due to exhaustive search was considerably more than the effect of examining all overlapping approximate patterns.

7.2 Regulatory Motifs in ChIP-on-chip Yeast Data

The results of applying the *LSHCS* algorithm on the ChIP-on-chip data set for the 21 transcription factors for which the six motif-finding algorithms failed to find a significant motif are shown in Table 7.

Parameter	*nPC*	*nCC*
$L = 8, d = 1$	0.037	0.063
$L = 8, d = 3$†	...	...
$L = 12, d = 2$	0	-0.009
$L = 12, d = 3$	0.032	0.0499
$L = 12, d = 4$†	...	...
$L = 16, d = 3$	0	-0.011
$L = 16, d = 4$	0.020	0.029
$L = 16, d = 5$†	...	...

Table 3: Performance of the *NoClust* pattern discovery method on the Drosophila genome using different input parameters (*nPC*=performance coefficient, *nCC*=correlation coefficient, *L*=pattern length, *d*=maximum Hamming distance allowed in pattern). † Cases where the algorithm did not terminate in 24 hours.

Parameter	*nPC*	*nCC*
$L = 8, d = 1$	0.039	0.065
$L = 8, d = 2$	0.017	0.015
$L = 8, d = 3$†	...	...
$L = 12, d = 2$	0.006	0.005
$L = 12, d = 3$	0.025	0.036
$L = 12, d = 4$†	...	...
$L = 16, d = 3$	0	-0.011
$L = 16, d = 4$	0.009	0.008
$L = 16, d = 5$†	...	...

Table 4: Performance of the *ExhSearch* pattern discovery method on the Drosophila genome using different input parameters (*nPC*=performance coefficient, *nCC*=correlation coefficient, *L*=pattern length, *d*=maximum Hamming distance allowed in pattern). † Cases where the algorithm did not terminate in 24 hours.

Parameter	Average *nPC*	Average *nCC*
LSHCS	0.019	0.024
NoClust	0.021	0.031
ExhSearch	0.016	0.020

Table 5: Comparison of *LSHCS*, *NoClust* and *ExhSearch* by summarizing results from parameter selections where all three algorithms terminated within 24 hours (*nPC*=performance coefficient, *nCC* =correlation coefficient, *L*=pattern length, *d*=maximum Hamming distance allowed in pattern).

Parameter	*LSHCS* Time	*NoClust* Time	*ExhSearch* Time
$L = 8, d = 1$	5:01	15:40	3:01:22
$L = 8, d = 2$	5:53	1:06:06	3:49:08
$L = 8, d = 3$†	4:13	...	...
$L = 12, d = 2$	18:14	35:54	17:05:37
$L = 12, d = 3$	24:48	2:33:12	17:27:43
$L = 12, d = 4$†	27:30	...	...
$L = 16, d = 3$	31:08	29:59	18:21:15
$L = 16, d = 4$	32:20	2:59:14	18:04:59
$L = 16, d = 5$†	24:15	...	...

Table 6: Time taken for *LSHCS*, *NoClust* and *ExhSearch* pattern discovery methods on the Drosophila genome using different input parameters. † Cases where one or more of the algorithm did not terminate in 24 hours.

Transcription Factor	Motif Found
ADR1	...
DAL80	Yes
GAL80	...
GCR1	...
GZF3	...
HAP2	...
HAP3	Yes
HAP5	...
MAC1	Yes
MET31	...
MET32	...
MOT3	Yes
MSN4	...
PUT3	...
RGT1	...
RLM1	...
ROX1	Yes
RTG3	...
SKO1	...
YAP6	Yes
YOX1	...

Table 7: Results of *LSHCS* on the ChIP-on-chip data set for transcription factors (TF) where common motif-finding algorithms failed to find a significant motif.

Pattern (Centroid)	Rank Sum P-Value	C-statistic
ABCCDFGJ	0.025	0.71
FFJJJJCC	0.004	0.70

Table 8: Statistical significance of approximate patterns found on a training set of 765 post-NSTEACS patients (15 deaths over a 90 day follow-up period) when evaluated on a test population of 250 patients (10 deaths).

In 6 of the 21 experiments conducted, the discriminative motif with the lowest p-value found by the *LSHCS* algorithm corresponded to the consensus motif in the literature, but was not found by any of the six motif-finding algorithms evaluated earlier. *LSHCS*, as well as the other six motif-finding algorithms, failed to find the discriminative motif in the remaining 15 cases although motifs are described in the literature.

7.3 Predictive Morphology Variations in Electrocardiogram Signals

Our pattern discovery method returned 2 approximate patterns that were assessed to have discriminative value in the training set (i.e., a C-statistic of more than 0.7 and a p-value of less than 0.05 after accounting for the Bonferroni correction). Representing the symbols obtained using SAX by the letters *A*-*J*, where *A* corresponds to the symbol class for the least beat-to-beat change in morphology and *J* denotes the symbol for the greatest change, the centroids for the approximate pattern can be written as:

$$ABCCDFGJ$$

$$FFJJJJCC$$

The first of these patterns is equivalent to increasing time-aligned energy changes between successive beats. This may suggest increased instability in the conduction system of the heart. The second pattern corresponds to a run of instability followed by a return to baseline. This pattern can be interpreted as a potential arrhythmia.

The results of testing both patterns on previously unseen data from 250 patients (with 10 deaths over a 90 day follow-up period) are shown in Table 8. Both patterns found by our approximate pattern discovery algorithm showed statistical significance in predicting death according to the C-statistic and rank sum criteria.

A comparison of the running times for the *LSHCS* and *NoSeqStats* algorithms is presented in Table 9. While the outputs produced by both algorithms were identical, the use of sequential statistics helped our *LSHCS* method decrease the runtime of the search process to almost half of what we encountered for the variation not using the methods proposed in Section 5.3. We also note that the *NoSeqStats* variation used considerably more memory than the *LSHCS* approach. This effect was due to *LSHCS* purging state for patterns that did not obey the inequality in Equation 9. In the absence of sequential statistics, *NoSeqStats* had to retain ranking information for all patterns till the training data set was completely analyzed.

Algorithm	Time
LSHCS	5:08:24
NoSeqStats	9:43:09

Table 9: Time taken by the *LSHCS* and *NoSeqStats* pattern discovery algorithms on the cardiovascular training data set.

8. Summary and Discussion

This paper presents an approach to efficiently learn patterns in labeled sequences that can be used for classification. We define patterns as groups of approximately matching subsequences, that is, all variations of a subsequence within a given Hamming radius. Our method represents a fully automated approach for unsupervised pattern discovery, where the goodness of patterns is assessed by measuring differences in their frequency of occurrence in positive and negative training examples.

We designed our pattern discovery algorithm to make it both accurate and efficient in terms of space and computation. We briefly review the central ideas of our work.

First, we include patterns from both positive and negative training sets in our search for discriminative activity. In many applications, we can consider positive examples as being associated with some physical phenomenon. Patterns that occur mainly in positive examples can be considered as potentially regulatory activity that may cause the phenomenon being studied. Conversely, patterns that occur mainly in negative examples (i.e., are absent in positive examples) can be considered as potentially supressive activity. The absence of these suppressive patterns in positive examples may promote the observed phenomenon. We allow for the discovery of both types of activity. This approach has a further advantage in that it reduces the need for prior knowledge. In the absence of negative examples, activity in positive training samples must be assessed through some assumption of statistical over-representation. By being able to directly compare positive examples against negative ones, we attempt to remove the need for such assumptions.

Second, to efficiently search for approximate patterns, we make use of locality sensitive hashing. LSH has been proposed as a randomized approximation algorithm to solve the nearest neighbor problem. We employ an iterative LSH method that is able to efficiently find groups of matching subsequences for subsequent analysis as candidate patterns.

Third, we explore the idea of clustering together subsequences, so that the number of candidate patterns can be reduced. This abstraction is intended to decrease the redundancy associated with evaluating a large number of approximate patterns with significant overlap. Reducing this redundancy, while still spanning the search space, provides an improvement in efficiency and also allows for more clarity in interpreting the results of the search process.

Finally, we make use of non-parametric statistical methods that have been designed to identify patterns with different distributions in both positive and negative examples. An extension of this work is the use of sequential methods, which only use as much of the training data as is needed to make a decision about a candidate pattern.

We evaluated the use of our pattern discovery algorithm on data from different real-world applications. On nucleotide sequences from the Drosophila genome, our method was able to discover approximate binding sites that were preserved upstream of genes. A similar result was seen for ChIP-on-chip data from the yeast genome. For cardiovascular data from patients admitted with

acute coronary syndromes, our pattern discovery approach identified approximately conserved sequences of morphology changes that were predictive of future death in a test population. Our data showed that the use of LSH, clustering, and sequential statistics improved the running time of the search algorithm by an order of magnitude without any noticeable effect on accuracy. These results suggest that our methods may allow for an efficient unsupervised approach to learn interesting dissimilarities between positive and negative examples, which may have a functional role.

Acknowledgments

We would like to thank Dorothy Curtis for helping with the computational needs for this project, Murtaza Zafar and Ebad Ahmed for their technical input regarding sequential statistics, and Manolis Kellis for suggesting the genomic data set for validation. We also thank the editors and anonymous reviewers for their feedback.

This work was supported by the Center for Integration of Medicine and Innovative Technology (CIMIT), the Harvard-MIT Division of Health Sciences and Technology (HST), the Industrial Technology Research Institute (ITRI), and Texas Instruments (TI).

References

Timothy L. Bailey and Charles Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *International Conference on Intelligent Systems in Molecular Biology*, pages 28–36, 1994.

Yosef Barash, Gill Bejerano, and Nir Friedman. A simple hyper-geometric approach for discovering putative transcription factor binding sites. *Algorithms in Bioinformatics*, pages 278–293, 2001.

Serafim Batzoglou, Lior Pachter, Jill P. Mesirov, Bonnie Berger, and Eric S. Lander. Human and mouse gene structure: comparative analysis and its application to exon prediction. *Genome Research*, pages 950–958, 2000.

Martin Bland and Douglas G. Altman. Multiple significance tests: the bonferroni method. *British Medical Journal*, page 170, 1995.

Alvis Brazma, Inge Jonassen, Ingvar Eidhammer, and David Gilber. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, pages 279–305, 1998.

Jeremy Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, pages 419–428, 2001.

Jeremy Buhler and Martin Tompa. Finding motifs using random projections. *Journal of Computational Biology*, pages 225–242, 2002.

Moises Burset and Roderic Guigo. Evaluation of gene structure prediction programs. *Genomics*, pages 353–367, 1996.

Stuart Daw, Charles E. Finney, and Eugene R. Tracy. A review of symbolic analysis of experimental data. *Review of Scientific Instruments*, pages 915–930, 2003.

Arthur L. Delcher, Simon Kasif, Robert D. Fleischmann, Jeremy Peterson, Owen White, and Steven L. Salzberg. Alignment of whole genomes. *Nucleic Acids Research*, pages 2369–2376, 1999.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

Eleazar Eskin and Pavel Pevzner. Finding composite regulatory patterns in dna sequences. *Bioinformatics*, pages 354–363, 2002.

Jean D. Gibbons. *Nonparametric Statistical Inference*. Marcel Dekker, 1985.

William N. Grundy, Timothy L. Bailey, Charles P. Elkan, and Michael E. Baker. Meta-meme: motif-based hidden markov models of protein families. *Computer Applications in the Biosciences*, pages 397–406, 1997.

Jaiwei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kauffman, 2005.

James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, pages 29–36, 1982.

Christopher T. Harbison, Benjamin Gordon, Tong I. Lee, Nicola J. Rinaldi, Kenzie D. Macisaac, Timothy W. Danford, Nancy M. Hannett, Jean-Bosco Tange, David B. Reynoalds, Jane Yoo, Ezra G. Jennings, Julia Zeitlingen, Dmitry K. Pokholok, Manolis Kellis, Alex Rolfe, Ken T. Takusagawa, Eric S. Lander, David K. Gifford, Ernest Fraenkel, and Richard A. Young. Transcriptional regulatory code of a eukaryotic genome. *Nature*, pages 99–104, 2004.

Taher H. Haveliwala, Aristides Gionis, and Piotr Indyk. Scalable techniques for clustering the web. In *WebDB Workshop*, 2000.

Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, pages 180–184, 1985.

Myles Hollander and Douglas A. Wolfe. *Nonparametric Statistical Methods*. John Wiley and Sons, 1999.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Symposium on Theory of Computing*, pages 604–613, 1998.

Tommi Jaakkola, Mark Diekhans, and David Haussler. Using the fisher kernel method to detect remote protein homologies. In *Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 149–158, 1999.

Manolis Kellis, Nick Patterson, Bruce Birren, Bonnie Berger, and Eric S. Lander. Methods in comparative genomics: genome correspondence, gene identification and regulatory motif discovery. *Journal of Computational Biology*, pages 319–355, 2004.

Anders Krogh. Hidden markov models for labeled sequences. In *IEEE International Conference on Pattern Recognition*, pages 140–144, 1994.

Charles E. Lawrence, Stephen F. Altschul, Mark S. Boguski, Jun S. Liu, Andrew F. Neuwald, and John C. Wootton. Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. *Science*, pages 208–214, 1993.

Erich L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks*. McGraw-Hill, 1975.

Christina Leslie, Eleazar Eskin, Jason Weston, and William S. Noble. Mismatch string kernels for svm protein classification. *Advances in Neural Information Processing Systems*, 2003.

Henry Leung and Francis Chin. Finding motifs from all sequences with and without binding sites. *Bioinformatics*, pages 2217–2223, 2006.

Jiuyong Li, Ada W. Fu, Hongxing He, Jie Chen, Huidong Jin, Damien McAullay, Graham Williams, Ross Sparks, and Chris Kelman. Mining risk patterns in medical data. In *ACM Conference on Knowledge Discovery in Data*, pages 770–775, 2005.

Nan Li and Martin Tompa. Analysis of computational approaches for motif discovery. *Algorithms for Molecular Biology*, page 8, 2006.

Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2003.

Shirley Liu, Douglas L Brutlag, and Jun S. Liu. Bioprospector: discovering conserved dna motifs in upstream regulatory regions of co-expressed genes. In *Pacific Symposium on Biocomputing*, pages 127–138, 2001.

Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Very Large Data Bases*, pages 950–961, 2007.

Bamshad Mobasher, Namit Jain, Eui-Hong Han, and Jaideep Srivasta. Web mining: pattern discovery from world wide web transactions. *Department of Computer Science, University of Minnesota Technical Report TR-96-050*, 1996.

Giulio Pavesi, Giancarlo Mauri, and Graziano Pesole. An algorithm for finding signals of unknown length in dna sequences. *Bioinformatics*, pages 207–214, 2001.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

Pavel A. Pevzner and Sing-Hoi Sze. Combinatorial approaches to finding subtle signal in dna sequences. In *International Conference on Intelligent Systems for Molecular Biology*, pages 269–278, 2000.

Ravi M. Phatarfod and Aidan Sudbury. A simple sequential wilcoxon test. *Australian Journal of Statistics*, pages 93–106, 1988.

Lawrence R. Rabiner, Aaron E. Rosenberg, and Stephen E. Levinson. Considerations in dynamic time warping algorithms for discrete word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 575–582, 1978.

Emma Redhead and Timothy L. Bailey. Discriminative motif discovery in dna and protein sequences using the deme algorithm. *BMC Bioinformatics*, page 385, 2007.

Marion R. Reynolds. A sequential signed-rank test for symmetry. *The Annals of Statistics*, pages 382–400, 1975.

Michael J. Shaw, Chandrasekar Subramaniam, Gek W. Tan, and Michael E. Welge. Knowledge management and data mining for marketing. *Decision Support Systems*, pages 127–137, 2001.

Rahul Siddharthan, Eric D. Siggia, and Erik van Nimwegen. Phylogibbs: a gibbs sampling motif finder that incorporates phylogeny. *PLoS Computational Biology*, pages 534–556, 2005.

Saurabh Sinha and Martin Tompa. Ymf: a program for discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, pages 3586–3588, 2003.

Zeeshan Syed, John V. Guttag, and Collin M. Stultz. Clustering and symbolic analysis in cardiovascular signals: discovery and visualization of medically relevant patterns in long-term data using limited prior knowledge. *EURASIP Journal on Advances in Signal Processing*, 2007.

Zeeshan Syed, Benjamin M. Scirica, Collin M. Stultz, and John V. Guttag. Risk-stratification following acute coronary syndromes using a novel electrocardiographic technique to measure variability in morphology. In *Computers in Cardiology*, 2008.

Saeed Tavazoie, Jason D. Hughes, Michael J. Campbell, Raymond J. Cho, and George M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, pages 281–285, 1999.

Martin Tompa, Nan Li, Timothy L. Bailey, George M. Church, Bart D. Moor, Eleazer Eskin, Alexander V. Favorov, Martin C. Frith, Yutao Fu, James Kent, Vsevolod J. Makeev, Andrei A. Mironov, William S. Noble, Giulio Pavesi, Graziano Pesole, Mireille Regnier, Nicolas Simonis, Saurabh Sinha, Gert Thijs, Jacques V. Helden, Mathias Vandenbogaert, Zhiping Weng, Christopher Workman, Chun Ye, and Zhou Zhu. Assessing computational tools for the discovery of transcription factor binding sites. *Nature Biotechnology*, pages 137–144, 2005.

Jason T.L. Wang, Bruce A. Shapiro, and Dennis E. Shasha. *Pattern Discovery in Biomolecular Data: Tools, Techniques, and Applications*. Oxford University Press, 1999.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, pages 80–83, 1945.

Edgar Wingender, Peter Dietze, Holger Karas, and Rainer Knuppel. Transfac: a database on transcription factors and their dna binding sites. *Nucleic Acids Research*, pages 238–241, 1996.

Journal of Machine Learning Research 10 (2009) 1937-1953 Submitted 1/09; Revised 5/09; Published 8/09

Hybrid MPI/OpenMP Parallel Linear Support Vector Machine Training

Kristian Woodsend K.WOODSEND@ED.AC.UK
Jacek Gondzio J.GONDZIO@ED.AC.UK
School of Mathematics and Maxwell Institute for Mathematical Sciences
University of Edinburgh
The Kings Buildings
Edinburgh, EH9 3JZ, UK

Editors: Sören Sonnenburg, Vojtech Franc, Elad Yom-Tov and Michele Sebag

Abstract

Support vector machines are a powerful machine learning technology, but the training process involves a dense quadratic optimization problem and is computationally challenging. A parallel implementation of linear Support Vector Machine training has been developed, using a combination of MPI and OpenMP. Using an interior point method for the optimization and a reformulation that avoids the dense Hessian matrix, the structure of the augmented system matrix is exploited to partition data and computations amongst parallel processors efficiently. The new implementation has been applied to solve problems from the PASCAL Challenge on Large-scale Learning. We show that our approach is competitive, and is able to solve problems in the Challenge many times faster than other parallel approaches. We also demonstrate that the hybrid version performs more efficiently than the version using pure MPI.

Keywords: linear SVM training, hybrid parallelism, largescale learning, interior point method

1. Introduction

Support vector machines (SVMs) are powerful machine learning techniques for classification and regression. They were developed by Vapnik (1998), and are based on statistical learning theory. They have been applied to a wide range of applications, with excellent results, and so they have received significant interest.

Like many machine learning techniques, SVMs involve a training stage, where the machine learns a pattern in the data from a *training data set*, and a separate test or validation stage where the ability of the machine to correctly predict labels is evaluated using a previously unseen *test data set*. This process allows parameters to be adjusted towards optimal values, while guarding against overfitting.

The training stage for Support Vector Machines involves at its core a dense convex quadratic optimization problem (QP). Solving this optimization problem is computationally expensive, primarily due to the dense Hessian matrix. Solving the QP with a general-purpose QP solver would result in the time taken to scale cubically with the number of data points ($O(n^3)$). Such a complexity result means that, in practise, the SVM training problem cannot be solved by general purpose optimization solvers.

Several schemes have been developed where a solution is built by solving a sequence of small-scale problems, where only a few data points (an *active set*) are considered at a time. Examples include decomposition (Osuna et al., 1997) and sequential minimal optimization (Platt, 1999), and state-of-the-art software use these techniques. Active-set techniques work well when the data is clearly separable by a hyperplane, so that the separation into active and non-active variables is clear. With noisy data, however, finding a good separating hyperplane between the two classes is not so clear, and the performance of these algorithms deteriorates (Woodsend and Gondzio, 2007).

In addition, the active set techniques used by standard software are essentially sequential—they choose a small subset of variables to form the active set at each iteration, and this selection is based upon the results of the previous iteration. It is not clear how to efficiently implement such an algorithm in parallel, due to the large number of iterations required and the dependencies between each iteration and the next.

Few approaches have been developed for training SVMs in parallel, yet multiple-core computers are becoming the norm, and data sets are becoming ever larger. It is notable that of the 44 submissions to compete in the PASCAL Challenge on Large-scale Learning (Sonnenburg et al., 2008), only 3 entries were parallel methods.

Parallelization schemes so far proposed have involved splitting the training data to give smaller, separable optimization sub-problems which can be distributed amongst the processors. Dong et al. (2003) used a block-diagonal approximation of the kernel matrix to derive independent optimization problems. The resulting SVMs were used to filter out samples that were likely not to be support vectors. A SVM was then trained on the remaining samples, using the standard serial algorithm. Collobert et al. (2002) proposed a mixture of multiple SVMs where single SVMs are trained on subsets of the training set and a neural network is used to assign samples to different subsets.

Another approach is to use a variation of the standard SVM algorithm that is better suited to a parallel architecture. Tveit and Engum (2003) developed an exact parallel implementation of the Proximal SVM (Fung and Mangasarian, 2001), which classifies points by assigning them to the closest of two parallel planes. Compared to the standard SVM formulation, the single constraint is removed and the result is an unconstrained QP; this is substantially different from the linear SVM task set in the PASCAL Challenge.

There have only been a few parallel methods in the literature which train a standard SVM on the whole of the data set. We briefly survey the methods of Zanghirati and Zanni (2003), Graf et al. (2005), Durdanovic et al. (2007) and Chang et al. (2008).

The algorithm of Zanghirati and Zanni (2003) decomposes the SVM training problem into a sequence of smaller, though still dense, QP sub-problems. Zanghirati and Zanni implement the inner solver using a technique called variable projection method, which is able to work efficiently on relatively large dense inner problems, and is suitable for implementing in parallel. The performance of the inner QP solver was improved in Zanni et al. (2006).

In the cascade algorithm introduced by Graf et al. (2005), the SVMs are layered. The support vectors given by the SVMs of one layer are combined to form the training sets of the next layer. The support vectors of the final layer are re-inserted into the training sets of the first layer at the next iteration, until the global KKT conditions are met. The authors show that this feedback loop corresponds to standard SVM training.

The algorithm of Durdanovic et al. (2007), implemented in the Milde software, is a parallel implementation of the sequential minimal optimization. The objective function of the dual form (see Equation 3 below) is expressed in terms of partial gradients. Variables are selected to enter

the working set, based on the steepest descent direction, and whether the variables are free to move within their box constraints. A second working set method considers pairwise contributions. Very large data sets can be split across processors. When a variable z_i enters the working set, the owner processor broadcasts the corresponding data vector x_i. All nodes calculate kernel functions and update their portion of the gradient vector. Although many of the operations within an iteration are parallelizable, a very large number of sequential outer iterations are still required. The authors use a hybrid approach to parallelization similar to ours described below, involving a multi-core BLAS library, but its use is limited to Layer 1 and 2 operations.

Another family of approaches to QP optimization are based on interior point method (IPM) technology, which works by delaying the split between active and inactive variables for as long as possible. IPMs generally work well on large-scale problems, largely because the number of iterations tends to grow very slowly with the problem dimension. Unfortunately each iteration requires the solving of a large system of linear equations. A straight-forward implementation of the standard SVM dual formulation has a per iteration complexity of $O(n^3)$, and would be unusable for anything but the smallest problems. Several sequential implementations of IPMs for support vector machines address this difficulty (Ferris and Munson, 2003; Fine and Scheinberg, 2002; Woodsend and Gondzio, 2007). Returning to parallel implementations, Chang et al. (2008) use parallel IPM technology for the optimizer, and avoid the problem of inverting the dense Hessian matrix by generating a low-rank approximation of the kernel matrix using partial Cholesky decomposition with pivoting. The dense Hessian matrix can then be efficiently inverted implicitly using the low-rank approximation and the Sherman-Morrison-Woodbury (SMW) formula. Moreover, a large part of the calculations at each iteration can be distributed amongst the processors effectively. The SMW formula has been widely used in interior point methods; however, sometimes it runs into numerical difficulties. Fine and Scheinberg (2002) constructed data sets where an SMW-based algorithm required many more iterations to terminate, and in some cases stalled before achieving an accurate solution. They also showed that this situation arises in real-world data sets.

Most of the previous approaches (Durdanovic et al. 2007 is the exception) have considered the parallel computer system as a cluster of independent processors, communicating through a message passing scheme such as MPI (MPI-Forum, 1995). Advances in technology have resulted in systems where several processing cores have access to a single memory space, and such symmetric multiprocessing (SMP) architectures are becoming prevalent. OpenMP (OpenMP Architecture Review Board, 2008) has proven to work effectively on shared memory systems, while MPI can be used for message passing between nodes. It can also be used to communicate between processors within an SMP node, but it is not immediately clear that this is the most efficient technique.

Most high performance computing systems are now clusters of SMP nodes. On such hybrid systems, a combination of message passing between SMP nodes and shared memory techniques inside each node could potentially offer the best parallelization performance from the architecture, although previous investigations have revealed mixed results (Smith and Bull, 2001; Rabenseifner and Wellein, 2003). A standard approach to combining the two schemes involves OpenMP parallelization inside each MPI process, while communication between the MPI processes is made only outside of the OpenMP regions. Rabenseifner and Wellein (2003) refer to this style as "master-only".

In this paper, we propose a parallel linear SVM algorithm that adopts this hybrid approach to parallelization. It trains the SVM using the full data set, using an interior point method to give efficient optimization, and Cholesky decomposition to give good numerical stability. MPI is used to

communicate between clusters, while within clusters we take advantage of the availability of highly efficient OpenMP-based BLAS implementations. Data is distributed evenly amongst the processors. Our approach directly tackles the most computationally expensive part of the optimization, namely the inversion of the dense Hessian matrix, through providing an efficient implicit inverse representation. By exploiting the structure of the problem, we show how this can be parallelized with excellent parallel efficiency. The resulting implementation is significantly faster at SVM training than active set methods, and it allows SVMs to be trained on data sets that would be impossible to fit within the memory of a single processor.

The structure of the rest of this paper is as follows. Section 2 gives an outline of interior point method for optimizing quadratic programs. Section 3 provides a short description of support vector machines and the formulation we use. Then in Section 4 we describe our approach to training linear SVMs, exploiting the structure of the QP and accessing memory efficiently. Numerical performance results are given in Section 5. Section 6 contains some concluding remarks.

We now briefly describe the notation used in this paper. x_i is the attribute vector for the i^{th} data point, and it consists of the observation values directly. There are n observations in the training set, and m attributes in each vector x_i. X is the $m \times n$ matrix whose columns are the attribute vectors x_i associated with each point. The classification label for each data point is denoted by $y_i \in \{-1, 1\}$. The variables $w \in \mathbb{R}^m$ and $z \in \mathbb{R}^n$ are used for the primal variables ("weights") and dual variables (α in SVM literature) respectively, and $w_0 \in \mathbb{R}$ for the bias of the hyperplane. Scalars and column vectors are denoted using lower case letters, while upper case letters denote matrices. D, S, U, V, Y and Z are the diagonal matrices of the corresponding lower case vectors.

2. Interior Point Methods

Interior point methods represent state-of-the-art techniques for solving linear, quadratic and nonlinear optimization programmes. In this section the key issues of implementation for QPs are discussed very briefly; for more details, see Wright (1997).

For the purposes of this paper, we need a method to solve the box and equality-constrained convex quadratic problem

$$\begin{aligned} \min_z \quad & \frac{1}{2} z^T Q z + c^T z \\ \text{s.t.} \quad & Az = b \\ & 0 \le z \le u, \end{aligned}$$

where u is a vector of upper bounds, and the constraint matrix A is assumed to have full row rank. Dual feasibility requires that $A^T \lambda + s - v - Qz = c$, where λ is the Lagrange multiplier associated with the linear constraint $Az = b$ and $s, v > 0$ are the Lagrange multipliers associated with the lower and upper bounds of z respectively.

At each iteration, an interior point method makes a damped Newton step towards satisfying the primal feasibility, dual feasibility and complementarity product conditions,

$$\begin{aligned} ZSe &= \mu e \\ (U - Z)Ve &= \mu e, \end{aligned}$$

for a given $\mu > 0$. e is the vector of all ones. We follow a common practice in interior point literature and denote with a capital letter (Z, S, U, V) a diagonal matrix with elements of the corresponding

vector (z,s,u,v) on the diagonal. The algorithm decreases μ before making another iteration, and continues until both infeasibilities and the duality gap (which is proportional to μ) fall below required tolerances.

The Newton system to be solved at each iteration can be transformed into the *augmented system equations*:

$$\begin{bmatrix} -(Q+\Theta^{-1}) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} r_c \\ r_b \end{bmatrix}, \tag{1}$$

where $\Delta z, \Delta\lambda$ are components of the Newton direction in the primal and dual spaces respectively, $\Theta^{-1} \equiv Z^{-1}S + (U-Z)^{-1}V$, and r_c and r_b are appropriately defined residuals. If the block $(Q+\Theta^{-1})$ is diagonal, an efficient method to solve such a system is to form the Schur complement $C = A(Q+\Theta^{-1})^{-1}A^T$, solve the smaller system $C\Delta\lambda = r_b + A(Q+\Theta^{-1})^{-1} r_c$ for $\Delta\lambda$, and back-substitute into (1) to calculate Δz. Unfortunately, as we shall see in the next section, for the case of SVM training the Hessian matrix Q is a completely dense matrix.

3. Support Vector Machines

In this section we briefly outline the standard SVM binary classification primal and dual formulations, and summarise how they can be reformulated as a separable QP (for more details, see Woodsend and Gondzio, 2007).

A Support vector machine (SVM) is a classification learning machine that learns a mapping between the features and the target label of a set of data points known as the *training set*, and then uses a hyperplane $w^T x + w_0 = 0$ to separate the data set and predict the class of further data points. The labels are the binary values "yes" or "no", which we represent using the values $+1$ and -1. The objective is based on the *structural risk minimization* principle, which aims to minimize the risk functional with respect to both the empirical risk (the quality of the approximation to the given data, by minimising the misclassification error) and maximize the confidence interval (the complexity of the approximating function, by maximising the separation margin) (Vapnik, 1998).

For a *linear kernel*, the attributes in the vector x_i for the i^{th} data point are the observation values directly, while for a *non-linear kernel* the observation values are transformed by means of a (possibly infinite dimensional) non-linear mapping Φ.

Concentrating on the linear SVM classifier, and using a 2-norm for the hyperplane weights w and a 1-norm for the misclassification errors $\xi \in \mathbb{R}^n$, the QP that forms the core of training the SVM takes the form:

$$\begin{aligned} \min_{w,w_0,\xi} \quad & \frac{1}{2} w^T w + \tau e^T \xi \\ \text{s.t.} \quad & Y(X^T w + w_0 e) \geq e - \xi \\ & w, w_0 \text{ free}, \quad \xi \geq 0, \end{aligned} \tag{2}$$

where τ is a positive constant that parameterizes the problem.

Due to the convex nature of the problem, a Lagrangian function associated with (2) can be formulated, and the solution will be at the saddle point. Partially differentiating the Lagrangian function gives relationships between the primal variables (w, w_0 and ξ) and the dual variables ($z \in \mathbb{R}^n$) at optimality, and substituting these relationships back into the Lagrangian function gives the

standard dual problem formulation

$$\begin{array}{ll} \min\limits_{z} & \frac{1}{2} z^T Y X^T X Y z - e^T z \\ \text{s.t.} & y^T z = 0 \\ & 0 \le z \le \tau e. \end{array} \tag{3}$$

However, using one of the optimality relationships, $w = XYz$, we can rewrite the quadratic objective in terms of w. Consequently, we can state the classification problem (3) as a separable QP:

$$\begin{array}{ll} \min\limits_{w,z} & \frac{1}{2} w^T w - e^T z \\ \text{s.t.} & w - XYz = 0 \\ & y^T z = 0 \\ & w \,\text{free}, \quad 0 \le z \le \tau e. \end{array} \tag{4}$$

The Hessian is simplified to the diagonal matrix $Q = \text{diag}\left(\begin{bmatrix} e_m \\ 0_n \end{bmatrix} \right)$ where $e_m = (1, 1, \ldots, 1) \in \mathbb{R}^m$, while the constraint matrix becomes:

$$A = \begin{bmatrix} I_m & -XY \\ 0 & y^T \end{bmatrix} \in \mathbb{R}^{(m+1)\times(m+n)}. \tag{5}$$

As described in Section 2, the Schur complement,

$$\begin{aligned} C &\equiv A(Q + \Theta^{-1})^{-1} A^T \\ &= \begin{bmatrix} I_m + XY\Theta_z YX^T & -XY\Theta_z y \\ -y^T \Theta_z YX^T & y^T \Theta_z y \end{bmatrix} \in \mathbb{R}^{(m+1)\times(m+1)}, \end{aligned}$$

can be formed efficiently from such matrices and used to determine the Newton step. The operation of building the matrix C is of order $O(n(m+1)^2)$, while inverting the resulting matrix is an operation of order $O((m+1)^3)$. The formulation (4) is the basis of our parallel algorithm, where building matrix C is split between the processors. This approach is efficient if $n \gg m$ (as was true with all the Challenge data sets), since building C is the most expensive operation, but it would not be suitable for data sets with a large number of features and $m \gg n$.

4. Implementing the QP for Parallel Computation

To apply (4) to truly large-scale data sets, it is necessary to employ linear algebra operations that exploit the block structure of the formulation (Gondzio and Sarkissian, 2003; Gondzio and Grothey, 2007). Between clusters, the emphasis is on partitioning the linear algebra operations to minimize interdependencies between processors. Within clusters, the emphasis is on accessing memory in the most efficient manner.

The approach described below was implemented using the OOPS interior point solver (Gondzio and Grothey, 2007).[1] We should note here that, as the parallel track of the Challenge was focused on shared memory algorithms, our submission to the Challenge used only the techniques described in Section 4.2.

1. Our implementation is available for academic use at `http://www.maths.ed.ac.uk/ERGO/software.html`.

4.1 Linear Algebra Operations Between Nodes

We use the augmented system matrix $H = \begin{bmatrix} -Q-\Theta^{-1} & A^T \\ A & 0 \end{bmatrix}$ corresponding to system (4), where $Q = \text{diag}\begin{bmatrix} e_m \\ 0 \end{bmatrix}$, Θ was described in Section 2 and A is given by Equation (5). This results in H having a symmetric bordered block diagonal structure. We can break H into blocks:

$$H = \begin{bmatrix} H_1 & & & & A_1^T \\ & H_2 & & & A_2^T \\ & & \ddots & & \vdots \\ & & & H_p & A_p^T \\ A_1 & A_2 & \dots & A_p & 0 \end{bmatrix},$$

where $H_i = -(Q_i + \Theta_i^{-1})$ are actually diagonal and A_i result from partitioning the data set evenly across the p processors. Due to the "arrow-head" structure of H, a block-based Cholesky decomposition of the matrix $H = LDL^T$ will be guaranteed to have the structure:

$$H = \begin{bmatrix} L_1 & & & \\ & \ddots & & \\ & & L_p & \\ L_{A_1} & \dots & L_{A_p} & L_C \end{bmatrix} \begin{bmatrix} D_1 & & & \\ & \ddots & & \\ & & D_p & \\ & & & D_C \end{bmatrix} \begin{bmatrix} L_1^T & & & L_{A_1}^T \\ & \ddots & & \vdots \\ & & L_p^T & L_{A_p}^T \\ & & & L_C^T \end{bmatrix}.$$

Exploiting this structure allows us to compute the blocks L_i, D_i and L_{A_i} in parallel. Terms that form the Schur complement can be calculated in parallel but must then be gathered, and the corresponding blocks L_C and D_C computed serially. This requires the exchange of matrices of size $(m+1)\times(m+1)$ between processors.

$$H_i = L_i D_i L_i^T \quad \Rightarrow D_i = -(Q_i + \Theta_i^{-1}), L_i = I, \tag{6}$$

$$L_{A_i} = A_i L_i^{-T} D_i^{-1} = A_i H_i^{-1}, \tag{7}$$

$$C = -\sum_{i=1}^{p} A_i H_i^{-1} A_i^T, \tag{8}$$

$$= L_C D_C L_C^T. \tag{9}$$

Matrix C is a dense matrix of relatively small size $(m+1)\times(m+1)$, and the Cholesky decomposition $C = L_C D_C L_C^T$ is performed in the normal way on a single processor. It is possible that a coarse-grained parallel implementation of Cholesky decomposition could give better performance (Luecke et al., 1992), but we did not include this in our implementation as the time taken to perform the decomposition is negligible compared to computing C.

Once the representation $H = LDL^T$ above is known, we can use it to compute the solution of the system $H\begin{bmatrix} \Delta z \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} r_c \\ r_b \end{bmatrix}$ through back-substitution. $\Delta z'$, $\Delta\lambda'$ and $\Delta\lambda''$ are vectors used for

intermediate calculations, with the same dimensions as Δz and $\Delta\lambda$.

$$\Delta\lambda'' = L_C^{-1}(r_b - \sum_i^p L_{A_i} r_{c_i}), \tag{10}$$

$$\Delta\lambda' = D_C^{-1}\Delta\lambda'', \tag{11}$$

$$\Delta\lambda = L_C^{-T}\Delta\lambda', \tag{12}$$

$$\Delta z_i' = D_i^{-1} r_{c_i}, \tag{13}$$

$$\Delta z_i = \Delta z_i' - L_{A_i}^T \Delta\lambda. \tag{14}$$

For the formation of LDL^T, Equations (6) and (7) can be calculated on each processor individually. Outer products (8) are then calculated, and the results gathered onto a single master processor to form C; this requires each processor to transfer approximately $\frac{1}{2}(m+1)^2$ elements. The master processor performs the Cholesky decomposition of C (9). Each processor needs to calculate $L_{A_i} r_{c_i}$, which again can be performed without any inter-processor communication, and the results are gathered onto the master processor. The master processor then performs the calculations in Equations (10), (11) and (12) of the back-substitution. Vector $\Delta\lambda$ is broadcast to all processors for them to calculate Equations (13) and (14) locally.

4.2 Linear Algebra Operations Within Nodes

Within each node, the bulk of operations are due to the contribution of each processor $A_i H_i^{-1} A_i^T$ to the calculation of the Schur complement in (8), and to a lesser extent the calculation of L_{A_i} in (7).

The standard technology for dense linear algebra operations is the BLAS library. Much of the effort to produce highly efficient implementations of BLAS Layer 3 (matrix-matrix operations) have concentrated on the routine GEMM, for good reason: Kågström et al. (1998) showed that it is possible to develop an entire BLAS Layer 3 implementation based on a highly optimized GEMM routine and a small amount of BLAS Layer 1 and Layer 2 routines. Their approach focused on efficiently organizing the accessing of memory, both through structuring the data for locality and through ordering operations within the algorithm. Matrices are partitioned into *panels* (block rows or block columns) and further partitioned into blocks of a size that fits in the processor's cache, where access times to the data are much shorter. Herrero (2006) has pursued these concepts further, showing that it is possible to develop an implementation offering competitive performance without the need for hand-optimized routines.

Goto and van de Geijn (2008) have shown that another limiting factor is the process of looking up mappings in the page table between virtual and physical addresses of memory. A more efficient approach ensures that the mappings for all the required data reside in the Translation Look-aside Buffer, effectively a cache for the page table. In practice, the best way of achieving this is to recast the matrix-matrix multiplications as a sum of panel-panel multiplications, repacking each panel into a contiguous buffer. This is the approach implemented in GotoBLAS, the library used in our implementation.

To perform GEMM $C := AB + C$, the algorithm described in Goto and van de Geijn (2008) divides the matrices into panels and uses three optimized components.

1. Divide matrix B into block row panels. Each panel $B_{p\bullet}$ contains all the columns we need, but fewer rows than the original matrix B. As required, pack $B_{p\bullet}$ into a contiguous buffer.

2. Divide matrix A into block column panels $A_{\cdot p}$, so that the inner dimensions of $A_{\cdot p}$ and $B_{p\cdot}$ match. Further divide A into blocks A_{ip}. As required, pack block A_{ip} into a contiguous buffer, so that by the end it is transposed and in the L2 cache.

3. Considering each block A_{ip} in turn, perform the multiplication $C_{i\cdot} := A_{ip}B_{p\cdot} + C_{i\cdot}$, with $B_{p\cdot}$ brought into the cache in column strips.

Additionally, it is possible in a multi-core system to coordinate the packing of $B_{p\cdot}$ between the processors, avoiding redundancy and improving performance.

Similar techniques using panels and blocks can be applied to Cholesky factorization (Buttari et al., 2009), but again these are not included in our implementation as the factorization of C is a relatively small part of the algorithm.

Returning to the SVM training problem, by casting the main computation of our algorithm in terms of matrix-matrix multiplications, we can take advantage of the above improvements for a multi-threaded architecture:

1. Consider a subblock of the constraint matrix A, consisting of all rows and the number of columns around the same size as $m+1$. Call this A_i.

2. Calculate L_{A_i} for this subblock, using (7). This involves Layer 1 operations, but these can be vectorized by the compiler.

3. Calculate $C := C + L_{A_i}A_i^T$ using the GEMM algorithm described above.

The performance gain of this approach is investigated in the next section.

5. Performance

In this section we compare the hybrid OpenMP/MPI version of our software with one using only MPI, and also our implementation against three other parallel SVM solvers. Data sets are taken from the PASCAL Challenge on Large-scale Learning, and the sizes we used are shown in Table 1. Due to memory restrictions, we reduced the number of samples in the FD and DNA data sets. Additionally, the DNA data set was modified from categories to binary features, increasing m by a factor of 4. The data sets were converted into a simple feature representation in SVM-light format. The software was run on a cluster of quad-core 3GHz Intel Xeon processors, each with 2GB RAM. The GotoBLAS library was used for BLAS functions, with the number of OpenMP threads set to 4, to match the number of cores. We also used the LAM implementation of the MPI library.

To compare the hybrid approach (using the techniques described in Sections 4.1 and 4.2) with pure MPI (using Section 4.1 only), we used the data sets alpha to zeta. The results are shown in Figure 1. They consistently show that, although the pure MPI approach has better properties in terms of parallel efficiency, the hybrid approach is always computationally more efficient. We believe this is a result of the multi-core processor architecture. The cores are associated with relatively small local cache memories, and such an architecture demands a fine-grained parallelism where, to reduce bus traffic, an operation is split into tasks that operate on small portions of data (Buttari et al., 2009). OpenMP is better suited to this fine-grained parallelism.

We made a comparison with other parallel software PGPDT (Zanni et al., 2006), PSVM (Chang et al., 2008), and Milde (Durdanovic et al., 2007). All of them are able to handle nonlinear as well as linear kernels, unlike our implementation. Using a linear kernel in each case, the results are shown

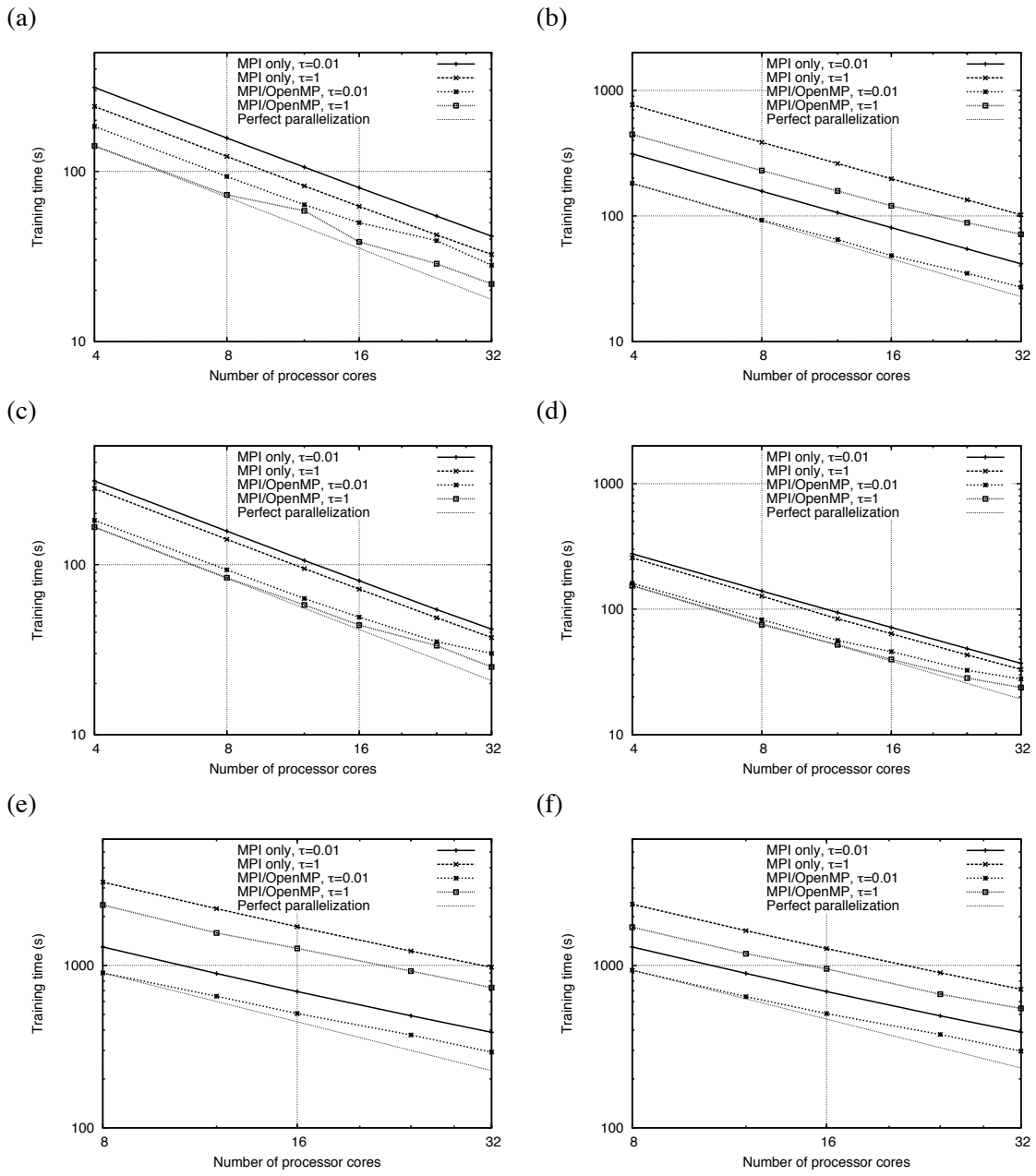


Figure 1: SVM training time with respect to the number of processors, for the PASCAL data sets (a) alpha, (b) beta, (c) gamma, (d) delta, (e) epsilon and (f) zeta. For each data set we trained using two values of τ. The results show that, although the pure MPI approach shows better parallel efficiency properties, the hybrid approach is always computationally more efficient.

Data Set	n	m
Alpha	500000	500
Beta	500000	500
Gamma	500000	500
Delta	500000	500
Espilon	500000	2000
Zeta	500000	2000
FD	2560000	900
OCR	3500000	1156
DNA	6000000	800

Table 1: PASCAL Challenge on Large-scale Learning data sets used in this paper.

Data Set	# cores	τ	OOPS	PGPDT	PSVM	Milde
Alpha	16	1	39	3673	1684	(80611)
		0.01	50	4269	4824	(85120)
Beta	16	1	120	5003	2390	(83407)
		0.01	48	4738	4816	(84194)
Gamma	16	1	44	—	1685	(83715)
		0.01	49	7915	4801	(84445)
Delta	16	1	40	—	1116	(57631)
		0.01	46	9492	4865	(84421)
Epsilon	32	1	730	—	17436	(58488)
		0.01	293	—	36319	(56984)
Zeta	32	1	544	—	14368	(22814)
		0.01	297	—	37283	(68059)
FD	32	1	3199	—	—	(39227)
		0.01	2152	—	—	(52408)
OCR	32	1	1361	—	—	(58307)
		0.01	1330	—	—	(36523)
DNA	48	1	2668	—	—	—
		0.01	6557	—	—	14821

Table 2: Comparison of parallel SVM training software on PASCAL data sets. Times are in seconds. In all cases except the DNA data set, the Milde software ran but did not terminate within 24 hours of runtime, so the numbers in brackets show when it was within 1% of its final objective value; — indicates that the software failed to load the problem.

in Table 2. With the exception of Milde (which has its own message passing implementation), the LAM implementation of the MPI library was used.

We required an objective value accuracy of $\varepsilon = 0.01$, and chose two values for τ within the range set in the Challenge, so we believe the training tasks are representative. In keeping with the evaluation method of the Challenge, the timings shown are for training and do not include time spent reading the data. The PSVM algorithm includes an additional partial Cholesky factorization

Data Set	τ	Parallel OOPS			Single processor OOPS		LibLinear		LaRank	
		n	cores	*t*	*n*	*t*	*n*	*t*	*n*	*t*
Alpha	1	500,000	16	39	500,000	122	500,000	147	500,000	3354
	0.01			50		151		112		2474
Beta	1	500,000	16	120	500,000	394	500,000	135	500,000	6372
	0.01			48		154		112		1880
Gamma	1	500,000	16	44	500,000	149	500,000	(8845)	500,000	—
	0.01			49		163		348.33		20318
Delta	1	500,000	16	40	500,000	137	500,000	(13266)	500,000	—
	0.01			46		134		429		—
Epsilon	1	500,000	32	730	210,000	951	250,000	316	500,000	5599
	0.01			293		374		265		2410
Zeta	1	500,000	32	544	230,000	1115	250,000	278	500,000	—
	0.01			297		449		248		—
FD	1	2,560,000	48	3199	500,000	1502	500,000	231	500,000	1537
	0.01			2152		840		193		332
OCR	1	3,500,000	32	1361	250,000	275	250,000	181	500,000	5695
	0.01			1330		297		121		4266
DNA	1	6,000,000	48	2668	600,000	175	600,000	144	600,000	300
	0.01			6557		176		30		407

Table 3: Comparison of our SVM training software OOPS, in parallel and on a single processor, with linear SVM software LibLinear and LaRank, again on PASCAL data sets. Times for training are in seconds. For the parallel software OOPS, each core had access to 2GB memory. The single processor codes had access to 4 cores and 8GB memory, and the larger data sets were reduced in size to fit. For LibLinear, brackets indicate that the iteration limit was reached. For LaRank, — indicates that the software did not terminate within 24 hours.

Data Set	OOPS	LibLinear	LaRank
Alpha	**0.1345**	0.1601	0.1606
Beta	**0.4988**	**0.4988**	0.5001
Gamma	**0.1174**	0.1185	0.1187
Delta	**0.1344**	0.1346	0.1355
Epsilon	**0.0341**	0.4935	0.4913
Zeta	**0.0115**	0.4931	0.4875
FD	**0.2274**	0.2654	0.3081
OCR	**0.1595**	0.1660	0.1681

Table 4: Accuracy measured using area under precision recall curve. These values are taken from the test results tables of the Evaluation pages of the PASCAL Challenge website.

procedure, which we also do not include in the training times. To make the training equivalent, the rank of the factorization was set to be the number of features m. The Milde software includes a number of termination criteria but not one based on the objective. Using the default criteria of maximum gradient below ε resulted in the software never terminating in all but one case within a 24

hour runtime limit. To be closer to the spirit of the Challenge, we show the time taken to be within 1% of the objective value at the end of 24 hours when the program was terminated prematurely, although in many cases the output indicated that the method was not yet converging.

The results show that our approach described in this paper and implemented in OOPS is typically one to two orders of magnitude faster than the other parallel SVM solvers, terminates reliably, and training times are reasonably consistent for different values of τ.

Unfortunately there were no other linear SVM implementations in the Parallel track of the PASCAL Challenge. Instead, we show in Table 3 training time results for two linear SVM codes that did participate in the PASCAL Challenge: LibLinear (Fan et al., 2008) which won the linear SVM track, and LaRank (Bordes et al., 2007) specialised for linear SVMs. Both codes ran serially, using the memory of 4 processor cores (8GB RAM in total), although it should be noted that as we used the GotoBLAS library when building LibLinear, it was able to use all four processor cores during BLAS operations. To make training possible, it was necessary to reduce the size of the larger data sets from the sizes given in Table 1; the number of samples used each time are shown as n in Table 3.

The presentation of the results is slightly unusual in that training times are not directly connected to accuracy results against a test set. This is because labelled validation and test data sets have not been made publicly available. Performance statistics related to the precision recall curve were evaluated on the Challenge website, and so instead we reproduce in Table 4 the results from the website for area under the precision recall curve results for the test data set. In general the precision of our method is consistent with the best of the other linear SVM methods that participated.

Taking the results in Tables 3 and 4 together, it is clear that LibLinear in particular is a very efficient implementation, even in the cases Alpha to Delta where it is working with the full data set. Table 4 clearly indicates that our approach consistently finds a high-quality solution to the separating hyperplane, measured in terms of classification accuracy, whereas for LibLinear the quality of the solution is lower. In many cases, the reduction in prediction accuracy is only small. The results in Table 4 for the Epsilon and Zeta data sets in particular show, however, that for some problems the quality of the solution from OOPS can be substantially better.

6. Conclusions

In this paper, we have shown how to develop a hybrid parallel implementation of linear Support Vector Machine training. The approach allows the entire data set to be used, and consists of the following steps:

1. Reformulating the problem to remove the dense Hessian matrix.
2. Using interior point method to solve the optimization problem in a predictable time, and Cholesky decomposition to give good numerical stability of implicit inverses.
3. Exploiting the block structure of the augmented system matrix, to partition the data and linear algebra computations amongst parallel processing nodes efficiently.
4. Within SMP nodes, casting the main computations as matrix-matrix multiplication where possible, partitioning the matrices to obtain better data locality, and using highly efficient BLAS implementation for a multi-threaded architecture.

The above steps were implemented in OOPS. Our results show that, for all cases, the hybrid implementation was faster than one using purely MPI, even though the MPI version had better parallel

efficiency. We used the hybrid implementation to solve very large problems from the PASCAL Challenge on Large-scale Learning, of up to a few million data samples. On these problems the approach described in this paper was highly competitive, and showed that even on data sets of this size, training times in the order of minutes are possible.

In this paper we have focused on linear kernels. It is possible to extend the techniques described above to handle non-linear kernels, by approximating the positive semidefinite kernel matrix K with a low-rank outer product representation such as partial Cholesky factorization $LL^T \approx K$ (Fine and Scheinberg, 2002). This approach produces the first r columns of the matrix L (corresponding to the r largest pivots) and leaves the other columns as zero, giving an approximation of the matrix K of rank r. Extending the work of Fine and Scheinberg, the diagonal $D \in \mathbb{R}^{n\times n}$ of the residual matrix $(K - LL^T)$ can be determined at no extra expense and included in a separable formulation for non-linear kernels:

$$\begin{aligned} \min_{w,z} \quad & \frac{1}{2}(w^T w + z^T D z) - e^T z \\ \text{s.t.} \quad & w - (YL)^T z = 0 \\ & -y^T z = 0 \\ & 0 \le z \le \tau e. \end{aligned}$$

Chang et al. (2008) describe how to perform partial Cholesky decomposition in a parallel environment. Data is segmented between the processors. All diagonal elements are calculated to determine pivot candidates. Then, for each of the r columns, the largest diagonal element is located. The corresponding pivot row of L and the original features need to be known by all processors, so this information is broadcast by the owner processor. With this information, all processors can update the section of the new column of L for which they are responsible, and also update corresponding diagonal elements. Although the algorithm requires the processors to be synchronised at each iteration, little of the data needs to be shared amongst the processors: the bulk of the communication between processors is limited to a vector of length m and a vector of at most length r. Note that matrix K is known only implicitly, through the kernel function, and calculating its values is an expensive process. The algorithm therefore calculates each kernel element required to form L only once, giving a complexity of $O(nr^2 + nmr)$ for the initial Cholesky factorization, and $O(nr^2 + r^3)$ for each IPM iteration of our algorithm.

The method described in this paper requires all sample data to be loaded into memory, and this clearly has an impact on the size of problem that can be tackled. It is possible to improve data handling and increase the storage capacity somewhat, for instance storing the data compactly and expanding sections into floating point numbers when needed by the BLAS routines (Durdanovic et al., 2007), but the scaling is still $O(nm^2)$. The direction of our further research is to develop methods that are able to safely ignore or remove data points from consideration as the algorithm progresses. In conjunction with exploiting the structure of the optimization problem as described in this paper, we believe this will offer further significant improvements to the overall training time.

Acknowledgments

The authors would like to thank the PASCAL organisers for the Challenge, the anonymous referees for their comments, Professor Zanni for providing the PGPDT software, and Dr Durdanovic for assistance with the Milde software. This work has made use of the resources provided by the

Edinburgh Compute and Data Facility (ECDF, http://www.ecdf.ed.ac.uk/). The ECDF is partially supported by the eDIKT initiative (http://www.edikt.org).

References

Antoine Bordes, Léon Bottou, Patrick Gallinari, and Jason Weston. Solving multiclass support vector machines with LaRank. In Zoubin Ghahramani, editor, *Proceedings of the 24th International Machine Learning Conference*, pages 89–96, Corvallis, Oregon, 2007. OmniPress. URL `http://leon.bottou.org/papers/bordes-2007`.

Alfredo Buttari, Julien Langou, Jakub Kurzak, and Jack Dongarra. A class of parallel tiled linear algebra algorithms for multicore architectures. *Parallel Computing*, 35(1):38–53, January 2009.

Edward Chang, Kaihua Zhu, Hao Wang, Hongjie Bai, Jian Li, Zhihuan Qiu, and Hang Cui. Parallelizing support vector machines on distributed computers. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 257–264, Cambridge, MA, 2008. MIT Press.

Ronan Collobert, Samy Bengio, and Yoshua Bengio. A parallel mixture of SVMs for very large scale problems. *Neural Computation*, 14(5):1105–1114, 2002.

Jian-xiong Dong, Adam Krzyzak, and Ching Suen. A fast parallel optimization for training support vector machine. In P. Perner and A. Rosenfeld, editors, *Proceedings of 3rd International Conference on Machine Learning and Data Mining*, pages 96–105, Leipzig, Germany, 2003. Springer Lecture Notes in Artificial Intelligence.

Igor Durdanovic, Eric Cosatto, and Hans-Peter Graf. Large scale parallel SVM implementation. In Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large Scale Kernel Machines*, chapter 5, pages 105–38. MIT Press, 2007.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Michael Ferris and Todd Munson. Interior point methods for massive support vector machines. *SIAM Journal on Optimization*, 13(3):783–804, 2003.

Shai Fine and Katya Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2002.

Glenn Fung and Olvi L. Mangasarian. Proximal support vector machine classifiers. In F. Provost and R. Srikant, editors, *Proceedings KDD-2001: Knowledge Discovery and Data Mining, August 26-29, 2001, San Francisco, CA*, pages 77–86, New York, 2001. Asscociation for Computing Machinery.

Jacek Gondzio and Andreas Grothey. Parallel interior point solver for structured quadratic programs: Application to financial planning problems. *Annals of Operations Research*, 152(1): 319–339, 2007.

Jacek Gondzio and Robert Sarkissian. Parallel interior point solver for structured linear programs. *Mathematical Programming*, 96(3):561–584, 2003.

Kazushige Goto and Robert A. van de Geijn. Anatomy of high-performance matrix multiplication. *ACM Trans. Math. Softw.*, 34(3):1–25, 2008.

Hans Peter Graf, Eric Cosatto, Léon Bottou, Igor Dourdanovic, and Vladimir Vapnik. Parallel support vector machines: the Cascade SVM. In Lawrence Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2005. volume 17.

José R. Herrero. *A Framework for Efficient Execution of Matrix Computations*. PhD thesis, Universitat Politècnica de Catalunya (UPC), July 2006.

Bo Kågström, Per Ling, and Charles van Loan. GEMM-based level 3 BLAS: high-performance model implementations and performance evaluation benchmark. *ACM Trans. Math. Softw.*, 24 (3):268–302, 1998.

Glenn R. Luecke, Jae H. Yun, and Philip W. Smith. Performance of parallel cholesky factorization algorithms using blas. *The Journal of Supercomputing*, 6(3):315–329, December 1992.

MPI-Forum. *MPI: A Message-Passing Interface Standard*. University of Tennessee, Knoxville, Tennessee, 1.1 edition, June 1995. URL `http://www.mpi-forum.org`.

OpenMP Architecture Review Board. *OpenMP Application Program Interface*, 3.0 edition, May 2008. URL `http://www.openmp.org`.

Edgar Osuna, Robert Freund, and Federico Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop*, pages 276–285. IEEE, 1997.

John Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher. J. C. Burges, and Alexander. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 185–208. MIT Press, 1999.

Rolf Rabenseifner and Gerhard Wellein. Comparison of parallel programming models on clusters of SMP nodes. In Hans Georg Bock, Ekaterina Kostina, Hoang Xuan Phu, and Rolf Rannacher, editors, *Modelling, Simulation and Optimization of Complex Processes: Proceedings of the International Conference on High Performance Scientific Computing, March 10-14, 2003, Hanoi, Vietnam*, pages 409–426. Springer, 2003.

Lorna Smith and Mark Bull. Development of mixed mode MPI / OpenMP applications. *Scientific Programming*, 2001.

Soeren Sonnenburg, Vojtech Franc, Elad Yom-Tov, and Michele Sebag. PASCAL large scale learning challenge, 2008. URL `http://largescale.first.fraunhofer.de/`.

Amund Tveit and Havard Engum. Parallelization of the incremental proximal support vector machine classifier using a heap-based tree topology. Technical report, IDI, NTNU, Trondheim, 2003.

Vladimir Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Kristian Woodsend and Jacek Gondzio. Exploiting separability in large-scale support vector machine training. Technical Report MS-07-002, School of Mathematics, University of Edinburgh, August 2007. Submitted for publication. Available at `http://www.maths.ed.ac.uk/~gondzio/reports/wgSVM.html`.

Stephen J. Wright. *Primal-Dual Interior-Point Methods*. S.I.A.M., 1997.

Gaetano Zanghirati and Luca Zanni. A parallel solver for large quadratic programs in training support vector machines. *Parallel Computing*, 29(4):535–551, 2003.

Luca Zanni, Thomas Serafini, and Gaetano Zanghirati. Parallel software for training large scale support vector machines on multiprocessor systems. *Journal of Machine Learning Research*, 7: 1467–1492, 2006.

Journal of Machine Learning Research 10 (2009) 1955-1988 Submitted 9/08; Revised 3/09; Published 8/09

Provably Efficient Learning with Typed Parametric Models

Emma Brunskill EMMA@CSAIL.MIT.EDU
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02143, USA

Bethany R. Leffler BLEFFLER@CS.RUTGERS.EDU
Lihong Li LIHONG@CS.RUTGERS.EDU
Michael L. Littman MLITTMAN@CS.RUTGERS.EDU
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA

Nicholas Roy NICKROY@MIT.EDU
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02143, USA

Editor: Sham Kakade

Abstract

To quickly achieve good performance, reinforcement-learning algorithms for acting in large continuous-valued domains must use a representation that is both sufficiently powerful to capture important domain characteristics, and yet simultaneously allows generalization, or sharing, among experiences. Our algorithm balances this tradeoff by using a stochastic, switching, parametric dynamics representation. We argue that this model characterizes a number of significant, real-world domains, such as robot navigation across varying terrain. We prove that this representational assumption allows our algorithm to be probably approximately correct with a sample complexity that scales polynomially with all problem-specific quantities including the state-space dimension. We also explicitly incorporate the error introduced by approximate planning in our sample complexity bounds, in contrast to prior Probably Approximately Correct (PAC) Markov Decision Processes (MDP) approaches, which typically assume the estimated MDP can be solved exactly. Our experimental results on constructing plans for driving to work using real car trajectory data, as well as a small robot experiment on navigating varying terrain, demonstrate that our dynamics representation enables us to capture real-world dynamics in a sufficient manner to produce good performance.

Keywords: reinforcement learning, provably efficient learning

1. Introduction

Reinforcement learning (RL) (Sutton and Barto, 1998) has had some impressive real-world successes, including model helicopter flying (Ng et al., 2004) and expert software backgammon players (Tesauro, 1994). Two of the key challenges in reinforcement learning are scaling up to larger, richer domains, and developing principled approaches for quickly learning to perform well. Our interest lies in developing algorithms for large continuous-valued environments, including problems such as

learning the best route to drive to work, or how a remote robotic rover can learn to traverse different types of terrain. To perform learning efficiently in such environments, we will assume that the world dynamics can be compactly described by a small set of simple parametric models, such as one for driving on highways and another for driving on small roads. We will prove that this assumption allows our algorithm to require an amount of experience that only scales polynomially with the state space dimension. We will also empirically demonstrate that these assumptions are realistic for several real world data sets, indicating that our Continuous-Offset Reinforcement Learning (CORL) algorithm may be well suited for large, high-dimensional domains.

A critical choice in the construction of a reinforcement-learning algorithm is how to balance between actions that gather information about the world environment (exploration) versus actions that are expected to yield high reward given the agent's current estimates of the world environment (exploitation). In early work, algorithms such as Q-learning were shown to perform optimally in the limit of infinite data (Watkins, 1989), but no finite-sample guarantees were known. More recently there have been three main branches of model-based reinforcement learning research concerned with the exploration problem. The first consists of heuristic approaches, some of which perform very well in practice, but lack performance guarantees (for example Jong and Stone 2007). The second branch strives to perform the action that optimally balances exploration and exploitation at each step. Such Bayesian approaches include the model parameters inside the state space of the problem. Poupart et al. (2006) assumed a fully observed discrete state space and modeled the underlying model parameters as hidden states, effectively turning the problem into a continuous-state partially observable Markov decision process (POMDP). Castro and Precup (2007) also assumed a fully observed discrete state space but represented the model parameters as counts over the different transitions and reward received, thereby keeping the problem fully observable. Doshi et al. (2008) considered a Bayesian approach for learning when the discrete state space is only partially observable, and Ross et al. (2008) considered learning in a partially-observed continuous-valued robot navigation problem. Approaches in the Bayesian RL framework run into inherent complexity problems and typically produce algorithms that only approximately solve their target optimality criteria.

In our work we will focus on achieving near optimality, making precise guarantees on when, and with what probability, it will be achieved. This type of approach to reinforcement learning was commenced by Kearns and Singh (2002) and Brafman and Tennenholtz (2002) who created algorithms that were guaranteed to achieve near optimal performance on all but a small number of samples, with high probability. We will refer to work in this line of research as "probably approximately correct" (PAC-MDP), as introduced by Strehl et al. (2006), and will discuss it further in the sections that follow. One of the appealing aspects of this area over a Bayesian RL approach is that it allows one to make precise statements about the efficiency and performance of algorithms: if the MDP or POMDP used in the Bayesian RL approach could be solved exactly with an informative prior, then this approach would likely outperform PAC-MDP approaches. However, when a Bayesian RL problem is only approximately solved or when the prior information is incorrect, it is unknown how far the resulting solution is from the optimal behavior. Our work lies within this third PAC-MDP approach, and draws upon the past advances made in this subfield, including our own initial work in this area (Brunskill et al., 2008). The current work makes a significant theoretical generalization of our initial results which requires different proof techniques, and presents a number of new experiments and discussions.

Within the PAC-MDP line of research, there has been little work on directly considering continuous-valued states. One exception is the work of Strehl and Littman (2008), who considered learning in continuous-valued state-action spaces. Their work assumed that a single dynamics representation was shared among all states, and that the noise parameter of the dynamics representation was known. The focus of their paper was slightly different than the current work, in that the authors presented a new online regression algorithm for determining when enough information was known to make accurate predictions.

An alternate approach to handling continuous state spaces is to discretize the space into a grid. This step enables prior PAC-MDP algorithms such as R-max (Brafman and Tennenholtz, 2002) to be applied directly to the discretized space. However, their representation of the world may not fully exploit existing structure. In particular, such a representation requires that the dynamics model for each state-action tuple is learned independently. Since each state-action can have entirely different dynamics, this approach has a great deal of representational power. However, as there is no sharing of dynamics information among states, it has a very low level of generalization. In contrast, the work of Strehl and Littman (2008) and the classic linear quadratic Gaussian regulator model (Burl, 1998) assume that the dynamics model is the same for all states, greatly restricting the representational power of these models in return for higher generalization and fast learning.

Recently, there have been several approaches that explore the middle ground of representational power and generalization ability. Jong and Stone (2007) assumed that the dynamics model between nearby states was likely to be similar, and used an instance-based approach to solve a continuous-state RL problem. Their experimental results were encouraging but no theoretical guarantees were provided, and the amount of data needed would typically scale exponentially with the state-space dimension. A stronger structural assumption is made in the work of Leffler et al. (2007), which focused on domains in which the discrete state space is divided into a set of types. States within the same type were assumed to have the same dynamics. The authors proved that a typed representation can require significantly less experience to achieve good performance compared to a standard R-max algorithm that learns each state-action dynamics model separately.

Our work draws on the recent progress and focuses on continuous-state, discrete-action, typed problems. By using a parametric model to represent the dynamics of each of a discrete set of types, we sacrifice some of the representational power of prior approaches (Leffler et al., 2007; Brafman and Tennenholtz, 2002) in return for improved generalization, but still retain a much more flexible representation than approaches that assume a single dynamics model that is shared across all states. In particular, we prove that restricting our representational power enables our algorithm to have a sample complexity that scales *polynomially* with the state-space dimension. An alternate approach is to place a uniformly spaced grid over the state space and solve the problem using the existing algorithms from Leffler et al. (2007) or Brafman and Tennenholtz (2002). However, this strategy results in an algorithm whose computational complexity scales exponentially with the state-space dimension.

Our algorithm involves a subroutine for solving a continuous-state MDP using the current model estimates. Outside of special cases like the linear Gaussian quadratic regulator problem (Burl, 1998), planning cannot be performed exactly for generic continuous-state MDPs. Therefore we explicitly incorporate the error introduced by approximate planning in our sample complexity bounds. This is in contrast to prior PAC-MDP approaches, which typically assume the estimated MDP can be solved exactly.

In particular, our dynamics representation is a simple noisy offset model, where the next state is presumed to be a function of the prior state, plus an offset and some Gaussian distributed noise. The offset and Gaussian parameters are assumed to be specified by the type t of the state and action a, thereby allowing all states of the same type to share dynamics parameters. More formally,

$$s' = s + \beta_{at} + \varepsilon_{at}, \tag{1}$$

where s is the current state, s' is the next state, $\varepsilon_{at} \sim \mathcal{N}(0, \Sigma_{at})$ is drawn from a zero-mean Gaussian with covariance Σ_{at}, and β_{at} is the offset.

In our experimental section we first demonstrate our algorithm on the standard RL PuddleWorld problem of Boyan and Moore (1995). We next illustrate the importance of learning the variance of different types by an example of an agent with a hard time deadline. The third example is a simulated decision problem in which an agent is trying to learn the best route for driving to work. The simulator uses real car-trajectory data to generate its trajectories. In the final experiment, a real robot car learns to navigate varying terrain. These experiments demonstrate that the noisy offset dynamics model, while simple, is able to capture real world dynamics for two different domains sufficiently adequately to allow the agent to quickly learn a good strategy.

At a high level, our work falls into the category of model-based reinforcement-learning algorithms in which the MDP model (Equation 1) can be *KWIK-learned* (Li et al., 2008; Li, 2009), and thus it is efficient in exploring the world. The Knows Whats It Knows (KWIK) framework is an alternate learning framework which incorporates characteristics of the Probably Approximately Correct (PAC) learning framework, which will be discussed further below, and the mistake bound framework. Though our theoretical development will follow a PAC-style approach, the KWIK framework provides another justification of the soundness and effectiveness of our algorithm.

The focus of this paper is on the sample complexity of the CORL algorithm. CORL assumes an approximate MDP planner to solve the current estimated MDP, and several such approximate planners with guarantees on the resulting solution involve a discretizaton that results in an exponential tiling of the state space. In such cases the computational complexity of CORL will scale exponentially with the number of dimensions. However, the experimental results demonstrate that CORL exhibits computational performance competitive with or better than existing approaches.

The rest of the paper proceeds as follows. In Section 2, we will briefly discuss the background to our work and then present the CORL algorithm. Section 3 presents our theoretical analysis of our algorithm. In Section 4 we present experimental results, and in Section 5 we conclude and discuss future work.

2. A Continuous-state Offset-dynamics Reinforcement Learner

This section introduces terminology and then presents our algorithm, CORL.

2.1 Background

The world is characterized by a continuous-state discounted MDP $M = \langle S, A, p(s'|s,a), R, \gamma \rangle$ where $S \subseteq \mathbb{R}^N$ is the N-dimensional state space, A is a set of discrete actions, $p(s'|s,a)$ is the transition dynamics, $\gamma \in [0,1)$ is the discount factor and $R : S \times A \to [0,1]$ is the reward function. In addition to the standard MDP formulation, each state s is associated with a single observable type $t \in T$. The total number of types is N_T and the mapping from states to types $S \to T$ is assumed to be known.

Algorithm 1 CORL

Input: N (dimension of the state space),$|A|$ (number of actions), N_T (number of types), R (reward model), γ (discount factor), N_{at} (minimum number of samples per state-action pair)
Set all type-action tuples $\langle t,a\rangle$ to be unknown and initialize the dynamics models (see text) to create an empirical known-type MDP model $\hat{M}_K$.
Start in a state s_0.
loop
  Solve MDP $\hat{M}_K$ using approximate solver and denote its optimal value function by Q_t.
  Select action $a = \text{argmax}_a Q_t(s,a)$.
  Increment the appropriate n_{at} count (where t is the type of state s).
  Observe transition to the next state s'.
  If n_{at} exceeds N_{at} then mark $\langle a,t\rangle$ as "known" and estimate the dynamics model parameters for this tuple.
end loop

The dynamics of the environment are determined by the current state type t and action a taken:

$$p(s'|s,a) = \mathcal{N}(s';s+\beta_{at},\Sigma_{at}).$$

Therefore, types partition the state space into regions, and each region is associated with a particular pair of dynamics parameters.

In this work, we focus on when the reward model is provided[1] and the dynamics model parameters are hidden. The parameters of the dynamics model, β_{at} and Σ_{at}, are assumed to be unknown for all types t and actions a at the start of learning. This model is a departure from prior related work (Abbeel and Ng, 2005; Strehl and Littman, 2008), which focuses on a more general linear dynamics model but assumes a single type and that the variance of the noise Σ_{at} is known. We argue that in many interesting problems, the variance of the noise is unknown and estimating this noise may provide the key distinction between the dynamics models of different types.

In reinforcement learning, the agent must learn to select an action a given its current state s. At each time step, it receives an immediate reward r based on its current state.[2] The agent then moves to a next state s' according to the dynamics model. The goal is to learn a policy $\pi : S \to A$ that allows the agent to choose actions to maximize the expected total reward it will receive. The value of a particular policy π is the expected discounted sum of future rewards that will be received from following this policy, and is denoted $V^\pi(s) = E_\pi[\sum_{j=0}^{\infty}\gamma^j r_j|s_0 = s]$, where r_j is the reward received on the j-th time step and s_0 is the initial state of the agent. Let π^* be the optimal policy, and its associated value function be $V^*(s)$.

2.2 Algorithm

Our algorithm (*c.f.*, Algorithm 1) is derived from the R-max algorithm of Brafman and Tennenholtz (2002). We first form a set of $\langle t,a\rangle$ tuples, one for each type-action pair. Note that each tuple

1. As long as the reward can be KWIK-learned (Li et al., 2008) then the results are easily extended to when the reward is unknown. KWIK-learnable reward functions include, for instance, Gaussian, linear and tabular rewards.
2. For simplicity, the reward is assumed to be only a function of state in this paper. It is straightforward to extend our results to the case when the reward function also depends on the action taken.

corresponds to a particular pair of dynamics model parameters, $\langle \beta_{at}, \Sigma_{at} \rangle$. A tuple is considered to be "known" if the agent has been in type t and taken action a a number N_{at} times. At each time step, we construct a new MDP $\hat{M}_K$ as follows, using the same state space, action space, and discount factor as the original MDP. If the number of times a tuple has been experienced, n_{at}, is greater than or equal to N_{at}, then we estimate the parameters for this dynamics model using maximum-likelihood estimation:

$$\tilde{\beta}_{at} = \frac{\sum_{i=1}^{n_{at}} (s'_i - s_i)}{n_{at}}, \quad (2)$$

$$\tilde{\Sigma}_{at} = \frac{\sum_{i=1}^{n_{at}} (s'_i - s_i - \tilde{\beta}_{at})(s'_i - s_i - \tilde{\beta}_{at})^T}{n_{at}} \quad (3)$$

where the sum ranges over all state-action pairs experienced for which the type of s_i was t, the action taken was a, and s'_i was the successor state. Note that while Equation 3 is a biased estimator, it is also popular and consistent, and becomes extremely close to the unbiased estimate when the number of samples n_{at} is large. We choose it because it makes our later analysis simpler.

Otherwise, we set the dynamics model for all states and the action associated with this type-action tuple to be a transition with probability 1 back to the same state. We also modify the reward function for all states associated with an unknown type-action tuple $\langle t_u, a_u \rangle$ so that all state-action values $Q(s_{t_u}, a_u)$ have a value of $V_{\max}$ (the maximum value possible, $1/(1-\gamma)$). We then seek to solve $\hat{M}_K$. This MDP includes switching dynamics with continuous states, and we are aware of no planners guaranteed to return the optimal policy for such MDPs in general. CORL assumes the use of an approximate solver to provide a solution for a MDP. There are a variety of existing MDP planners, such as discretizing or using a linear function approximation, and we will consider particular planner choices in the following sections. At each time step, the agent chooses the action that maximizes the estimate of its current approximate value according to Q_t: $a = \text{argmax}_a Q_t(s,a)$. The complete algorithm is shown in Algorithm 1.

3. Learning Complexity

In this section we will first introduce relevant background and then provide a formal analysis of the CORL algorithm.

3.1 Preliminaries and Framework

When analyzing the performance of an RL algorithm $\mathcal{A}$, there are many potential criteria to use. In our work, we will focus predominantly on sample complexity with a brief mention of computational complexity. Computational complexity refers to the number of operations executed by the algorithm for each step taken by the agent in the environment. We will follow Kakade (2003) and use *sample complexity* as shorthand for the *sample complexity of exploration*. It is the number of time steps at which the algorithm, when viewed as a non-stationary policy π, is not ε-optimal at the current state; that is, $Q^*(s,a) - Q^\pi(s,a) > \varepsilon$ where Q^* is the optimal state-action value function and Q^π is the state-action value function of the non-stationary policy π. Following Strehl et al. (2006), we are interested in showing, for a given ε and δ, that with probability at least $1-\delta$ the sample complexity of the algorithm is less than or equal to a polynomial function of MDP parameters. Note that we only consider the number of samples to ensure the algorithm will learn and execute a near-optimal

policy with high probability. As the agent acts in the world, it may be unlucky and experience a series of state transitions that poorly reflect the true dynamics due to noise.

Prior work by Strehl et al. (2006) provided a framework for analyzing the sample complexity of R-max-style RL algorithms. This framework has since been used in several other papers (Leffler et al., 2007; Strehl and Littman, 2008) and we will also adopt the same approach. We first briefly discuss the structure of this framework.

Strehl et al. (2006) defined an RL algorithm to be greedy if it chooses its action to be the one that maximizes the value of the current state s ($a = \text{argmax}_{a \in A} Q(s,a)$). Their main result goes as follows: let $\mathcal{A}(\varepsilon, \delta)$ denote a greedy learning algorithm. Maintain a list K of "known" state-action pairs. At each new time step, this list stays the same unless during that time step a new state-action pair becomes known. MDP $\hat{M}_K$ is the agent's current estimated MDP, consisting of the agent's estimated models for the known state-action pairs, and self loops and optimistic rewards (as in our construction described in the prior section) for unknown state-action pairs. MDP M_K is an MDP which consists of the true (underlying) reward and dynamics models for the known state-action pairs, and again self loops and optimistic rewards for the unknown state-action pairs. To be clear, the only difference between MDP $\hat{M}_K$ and MDP M_K is that the first uses the agent's experience to generate estimated models for the known state-action pairs, and the second uses the true model parameters. π is the greedy policy with respect to the current state-action values $Q_{\hat{M}_K}$ obtained by solving MDP $\hat{M}_K$: $V^{\pi}_{\hat{M}_K}$ is the associated value function for $Q_{\hat{M}_K}$ and may equivalently be viewed as the value of policy π computed using the estimated model parameters. $V^{\pi}_{M_K}$ is the value of policy π computed using the true model parameters. Assume that ε and δ are given and the following three conditions hold for all states, actions and time steps:

1. $Q^*(s,a) - Q_{\hat{M}_K}(s,a) \leq \varepsilon$.
2. $V^{\pi}_{\hat{M}_K}(s) - V^{\pi}_{M_K}(s) \leq \varepsilon$.
3. The total number of times the agent visits a state-action tuple that is not in K is bounded by $\zeta(\varepsilon, \delta)$ (the *learning complexity*).

Then, Strehl et al. (2006) show for any MDP M, $\mathcal{A}(\varepsilon, \delta)$ will follow a 4ε-optimal policy from its initial state on all but N_{total} time steps with probability at least $1 - 2\delta$, where N_{total} is polynomial in the problem's parameters $(\zeta(\varepsilon, \delta), \frac{1}{\varepsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma})$.

The majority of our analysis will focus on showing that our algorithm fulfills these three criteria. In our approach, we will define the known state-action pairs to be all those state-actions for which the type-action pair $\langle t(s), a \rangle$ is known. We will assume that the absolute values of the components in Σ_{at} are upper bounded by a known constant B_σ which is, without loss of generality, assumed to be greater than or equal to 1. This assumption is often true in practice. We denote the determinant of matrix D by $\det D$, the trace of a matrix D by $\text{tr}(D)$, the absolute value of a scalar d by $|d|$ and the p-norm of a vector v by $\|v\|_p$. Full proofs, when omitted, can be found in the Appendix.

3.2 Analysis

Our analysis will serve to prove the main result:

Theorem 1 *For any given δ and ε in a continuous-state noisy offset dynamics MDP with N_T types where the covariance along each dimension of all the dynamics models is bounded by $[-B_\sigma, B_\sigma]$,*

on all but N_{total} time steps, our algorithm will follow a 4ε-optimal policy from its current state with probability at least $1-2\delta$, where N_{total} is polynomial in the problem parameters $(N, |A|, N_T, \frac{1}{\varepsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma}, \frac{1}{\lambda_N}, B_\sigma)$ where λ_N is the smallest eigenvalue of the dynamics covariance matrices.

Proof To prove this, we need to demonstrate that the three criteria of Strehl et al. (2006) hold. The majority of our effort will focus on the second criterion. This criterion states that the value of states under the estimated known-state MDP $\hat{M}_K$ must be very close to the value of states under the known-state MDP M_K that uses the true model parameters for all known type-action pairs. To prove this we must bound how far away the model parameters estimated from the agent's experience can be from the true underlying parameters, and how this relates to error in the resulting value function. We must also consider the error induced by approximately solving the estimated MDP $\hat{M}_K$. Achieving a given accuracy level in the final value function creates constraints on how close the estimated model parameters must be to the true model parameters. We will illustrate how these constraints relate to the amount of experience required to achieve these constraints. This in turn will give us an expression for the number of samples required for a type-action pair to be known, or the learning complexity for our algorithm. Once we have proved the second criterion we will discuss how the other two conditions are also met.

Therefore we commence by formally relating how the amount of experience (number of transitions) of the agent corresponds to the accuracy in the estimated dynamics model parameters.

Lemma 2 *Given any $\varepsilon, \delta > 0$, then after $T = \frac{12N^2B_\sigma^2}{\varepsilon^2\delta}$ transition samples (s,a,s') with probability at least $1-\frac{2\delta}{3}$, the estimated offset parameter $\tilde{\beta}$, computed by Equation 2, and estimated covariance parameters $\tilde{\sigma}_{ij}$, computed by Equation 3, will deviate from the true parameters β and σ_{ij} by at most ε: $\Pr(\|\tilde{\beta}-\beta\|_2 \leq \varepsilon) \geq 1-\frac{\delta}{3}$ and $\Pr(\max_i |\tilde{\sigma}_{ij}-\sigma_{ij}| \leq \varepsilon) \geq 1-\frac{\delta}{3}$.*

Proof T will be the maximum of the number of samples to guarantee the above bounds for the offset parameter β and the number of samples needed for a good estimate of the variance parameter. We first examine the offset parameter:

Lemma 3 *Given any $\varepsilon, \delta > 0$, define $T_\beta = \frac{3N^2B_\sigma}{\varepsilon^2\delta}$. If there are T_β transition samples (s,a,s'), then with probability at least $1-\frac{\delta}{3}$, the estimated offset parameter $\tilde{\beta}$, computed by Equation 2, will deviate from the true offset parameter β by no more than ε along any dimension d; formally, $\Pr(\max_d \|\tilde{\beta}_d - \beta_d\|_2 \geq \frac{\varepsilon}{\sqrt{N}}) \leq \frac{\delta}{3N}$.*

Proof From Chebyshev's inequality, we know

$$P(|(s'_{id}-s_{id})-\beta_d| \geq \frac{\varepsilon}{\sqrt{N}}) \leq \frac{\sigma_d^2 N}{\varepsilon^2},$$

where s_{id} and σ_d^2 are the value of the i-th state and variance of the offset along dimension d, respectively. Using the fact that the variance of a sum of T_β i.i.d. variables is just T_β multiplied by the variance of a single variable, we obtain

$$\begin{aligned}
\Pr(|\sum_{i=1}^{T_\beta}(s'_{id}-s_{id}) - T_\beta\beta_d| \geq T_\beta\frac{\varepsilon}{\sqrt{N}}) &\leq \frac{T_\beta\sigma_d^2 N}{T_\beta^2\varepsilon^2} \\
\Pr(|\tilde{\beta}_d - \beta_d| \geq \frac{\varepsilon}{\sqrt{N}}) &\leq \frac{\sigma_d^2 N}{T_\beta\varepsilon^2}.
\end{aligned}$$

We require the right-hand side above be at most $\frac{\delta}{3N}$ and solve for T_β:

$$T_\beta = \frac{3\sigma_d^2 N^2}{\delta\varepsilon^2}.$$

We know that the variance along any dimension is bounded above by B_σ so we can substitute this in the above expression to derive a bound on the number of samples required:

$$T_\beta \geq \frac{3B_\sigma N^2}{\delta\varepsilon^2}.$$

■

Lemma 3 immediately implies a bound on the L_2 norm between the estimated offset parameter vector and the true offset parameter vector, as follows:

Lemma 4 *Given any* $\varepsilon, \delta > 0$, *if* $\Pr(\max_d |\tilde{\beta}_d - \beta_d| \geq \frac{\varepsilon}{\sqrt{N}}) \leq \frac{\delta}{3N}$, *then* $\Pr(\|\tilde{\beta} - \beta\|_2 \geq \varepsilon) \leq \frac{\delta}{3}$.

Proof By a union bound, the probability that any of the dimensions exceeds an estimation error of at most $\frac{\varepsilon}{\sqrt{N}}$ is at most $\frac{\delta}{3}$. Given this, with probability at least $1 - \frac{\delta}{3}$ all dimensions will simultaneously have an estimation error of less than $\frac{\varepsilon}{\sqrt{N}}$ and from the definition of the L2 norm this immediately implies that $\|\tilde{\beta} - \beta\|_2 \leq \varepsilon$. ■

We next analyze the number of samples needed to estimate the covariance accurately.

Lemma 5 *Assume* $\max_d |\tilde{\beta}_d - \beta_d| \leq \varepsilon$ *for* $\varepsilon < 1/4$. *Given any* $\delta > 0$, *define* $T_\sigma = \frac{12N^2 B_\sigma^2}{\delta\varepsilon^2}$. *If there are* T_σ *transition samples* (s, a, s'), *then with probability at most* $\frac{\delta}{3}$, *the estimated covariance parameter* $\tilde{\sigma}_{ij}$, *computed by Equation 3, deviates from the true covariance parameter* σ_{ij} *by more than* ε *over all entries* ij; *formally,* $\Pr(\max_{i,j} |\tilde{\sigma}_{ij} - \sigma_{ij}| \geq \varepsilon) \leq \frac{\delta}{3}$.

We provide the proof of Lemma 5 in the appendix: briefly, we again use Chebyshev's inequality which requires us to bound the variance of the sample covariance.

Combining Lemmas 4 and 5 gives a condition on the minimum number of samples necessary to ensure, with high probability, that the estimated parameters of a particular type-action dynamics model are close to the true parameters. Without loss of generality, assume $B_\sigma \geq 1$, then

$$T = \max\{T_\beta, T_\sigma\} = \max\left\{\frac{3N^2 B_\sigma}{\varepsilon^2\delta}, \frac{12N^2 B_\sigma^2}{\varepsilon^2\delta}\right\} = \frac{12N^2 B_\sigma^2}{\varepsilon^2\delta}.$$

■

From Lemma 2 we now have an expression that relates how much experience the agent needs in order to have precise estimates of each model parameter. We next need to establish the distance between two dynamics models which have different offset and covariance parameters. This distance will later be important for bounding the value function difference between the estimated model MDP $\hat{M}_K$ and the true model MDP M_K.

Following Abbeel and Ng (2005), we choose to use the variational distance between two dynamics models P and Q:

$$d_{var}(P(x),Q(x)) = \frac{1}{2}\int_{\mathcal{X}} |P(x)-Q(x)|dx.$$

In our algorithm, β_1 and Σ_1 are the true offset parameter and covariance matrix of the Gaussian distribution, and β_2 and Σ_2 are the offset parameter and covariance matrix estimated from data. Since we can guarantee that they can be made arbitrarily close (element-wise), we will be able to bound the variational distance between two Gaussians, one defined with the true parameters and the other with the estimated parameters. The real-valued, positive eigenvalues of Σ_1 and Σ_2 are denoted by $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N > 0$ and $\lambda'_1 \geq \lambda'_2 \geq \cdots \geq \lambda'_N > 0$, respectively. Because of symmetry and positive definiteness of Σ_1 and Σ_2, λ_i and λ'_i must be real as well as positive. Since all eigenvalues are positive, they are also the singular values of their respective matrices.

Lemma 6 *Assume* $\max_{i,j} |\Sigma_1(i,j) - \Sigma_2(i,j)| \leq \varepsilon$, *and* $N\varepsilon \left\|\Sigma_1^{-1}\right\|_\infty < 1$, *then,*

$$d_{var}(\mathcal{N}(s'-s|\beta_1,\Sigma_1),\mathcal{N}(s'-s|\beta_2,\Sigma_2)) \leq \frac{||\beta_1-\beta_2||_2}{\sqrt{\lambda_N}} + \sqrt{\frac{N^2\varepsilon}{\lambda_N} + \frac{2N^3 B_\sigma \varepsilon}{\lambda_N^2 - N^{1.5}\varepsilon\lambda_N}}.$$

Proof We will use $\mathcal{N}(\beta,\Sigma)$ as an abbreviation for $\mathcal{N}(s'-s|\beta,\Sigma)$. Then

$$\begin{aligned} d_{var}(\mathcal{N}(\beta_1,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_2)) &\leq d_{var}(\mathcal{N}(\beta_1,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_1)) + d_{var}(\mathcal{N}(\beta_2,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_2)) \\ &= \frac{||(\mathcal{N}(\beta_1,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_1))||_1}{2} + \frac{||(\mathcal{N}(\beta_2,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_2))||_1}{2} \\ &\leq \sqrt{2\mathrm{d_{KL}}(\mathcal{N}(\beta_1,\Sigma_1)\,\|\,\mathcal{N}(\beta_2,\Sigma_1))} + \sqrt{2\mathrm{d_{KL}}(\mathcal{N}(\beta_2,\Sigma_1)\,\|\,\mathcal{N}(\beta_2,\Sigma_2))} \end{aligned}$$

where $\mathrm{d_{KL}}(\|)$ is the Kullback-Leibler divergence. The first step follows from the triangle inequality and the last step follows from Kullback (1967) (included for completeness in Lemma 14 in the appendix).

The KL divergence between two N-variate Gaussians has the closed form expression

$$\mathrm{d_{KL}}(\mathcal{N}(\beta_1,\Sigma_1)\,\|\,\mathcal{N}(\beta_2,\Sigma_2)) = \frac{1}{2}\left((\beta_1-\beta_2)^T\Sigma_1^{-1}(\beta_1-\beta_2) + \ln\frac{\det\Sigma_2}{\det\Sigma_1} + \mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) - N\right).$$

Substituting this expression into the above bound on d_{var} we get

$$d_{var}(\mathcal{N}(\beta_1,\Sigma_1),\mathcal{N}(\beta_2,\Sigma_2)) \leq \sqrt{(\beta_1-\beta_2)^T\Sigma_1^{-1}(\beta_1-\beta_2)} + \sqrt{\ln\left(\frac{\det\Sigma_2}{\det\Sigma_1}\right) + \mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) - N}. \quad (4)$$

Our proof relies on bounding both terms of Equation 4. Note that this expression reduces (up to a constant) to the bound proved by Abbeel and Ng (2005) when the variance is known.

We now start with the first term of Equation 4:

Lemma 7

$$(\beta_1-\beta_2)^T\Sigma_1^{-1}(\beta_1-\beta_2) \leq \frac{1}{\lambda_N}||\beta_1-\beta_2||_2^2.$$

Proof First note that since Σ_1^{-1} is a Hermitian matrix,

$$\frac{(\beta_1-\beta_2)^T\Sigma_1^{-1}(\beta_1-\beta_2)}{||\beta_1-\beta_2)||_2^2}$$

is a Rayleigh quotient which is bounded by the maximum eigenvalue of Σ_1^{-1}. The eigenvalues of Σ_1^{-1} are precisely the reciprocals of the eigenvalues of Σ_1. Therefore, the Rayleigh quotient above is at most $\frac{1}{\lambda_N}$:

$$(\beta_1-\beta_2)^T\Sigma_1^{-1}(\beta_1-\beta_2) \leq \frac{||\beta_1-\beta_2||_2^2}{\lambda_N}.$$

■

We now provide lemmas that bound the components of the second term of Equation 4: proofs are provided in the appendix.

Lemma 8 *If* $\max_{i,j}|\Sigma_1(i,j)-\Sigma_2(i,j)| \leq \varepsilon$ *for any* $1 \leq i,j \leq N$, *then*

$$\left|\ln\frac{\det\Sigma_2}{\det\Sigma_1}\right| \leq N\varepsilon\left(\frac{1}{\lambda_1}+\frac{1}{\lambda_2}+\cdots+\frac{1}{\lambda_N}\right) \leq \frac{N^2\varepsilon}{\lambda_N}.$$

Lemma 9 *If* $\max_{i,j}|\Sigma_1(i,j)-\Sigma_2(i,j)| \leq \varepsilon$ *and* $N\varepsilon\left\|\Sigma_1^{-1}\right\|_1 < 1$, *then*

$$\mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right)-N \leq \frac{2N^3\varepsilon B_\sigma}{\lambda_N^2-(N)^{1.5}\lambda_N\varepsilon}.$$

Combining the results of Lemmas 7, 8, and 9 completes the proof of Lemma 6. ■

Note this bound is tight when the means and the variances are the same.

At this point we can relate the number of experiences (samples) of the agent to a distance measure between the estimated dynamics model (for a particular type-action) and the true dynamics model.

We now bound the error between the state-action values of the true MDP model M_K solved exactly and the approximate state-action values of our estimated model MDP $\hat{M}_K$ obtained using an approximate planner, as a function of the error in the dynamics model estimates. This is a departure from most related PAC-MDP work which typically assumes the existence of a planning oracle for choosing actions given the estimated model.

Lemma 10 ***(Simulation Lemma)*** *Let* $M_1=\langle S,A,p_1(\cdot|\cdot,\cdot),R,\gamma\rangle$ *and* $M_2=\langle S,A,p_2(\cdot|\cdot,\cdot),R,\gamma\rangle$ *be two MDPs*[3] *with dynamics as characterized in Equation 1 and non-negative rewards bounded above by 1. Given an* ε *(where* $0<\varepsilon\leq V_{\max}$*), assume that for all state-action tuples* (s,a), $d_{var}(p_1(\cdot|s,a),p_2(\cdot|s,a)) \leq (1-\gamma)^2\varepsilon/(2\gamma)$ *and the error incurred by approximately solving a MDP, defined as* ε_{plan} *is also at most* $(1-\gamma)^2\varepsilon/(2\gamma)$ *(to be precise,* $\varepsilon_{plan}=||V^*-\tilde{V}^*||_\infty \leq (1-\gamma)^2\varepsilon/(2\gamma)$ *where* $\tilde{V}^*$ *is the value computed by the approximate solver). Let* π *be a policy that can be applied to both* M_1 *and* M_2. *Then, for any stationary policy* π, *for all states* s *and actions* a, $|Q_1^\pi(s,a)-\tilde{Q}_2^\pi(s,a)| \leq \varepsilon$, *where* $\tilde{Q}_2^\pi$ *denotes the state-action value obtained by using an approximate MDP solver on MDP* M_2 *and* Q_1^π *denotes the true state-action value for MDP* M_1 *for policy* π.

3. For simplicity we present the results here without reference to types. In practice, each dynamics parameter would be subscripted by its associated MDP, type, and action.

Proof Let $\Delta_Q = \max_{s,a} |Q_1^\pi(s,a) - \tilde{Q}_2^\pi(s,a)|$ and define $\tilde{V}_2^\pi$ to be the approximate value of policy π computed using an approximate MDP solver on MDP M_2, and V_1^π be the exact value of policy π on MDP M_1. Note that since we are taking the max over all actions, Δ_Q is also equal to or greater than $\max_s |V_1^\pi(s) - \tilde{V}_2^\pi(s)|$. Let $Lp_2(s'|s,a)$ denote an approximate backup for MDP M_2.

Since these value functions are the fixed-point solutions to their respective Bellman operators, we have for every (s,a) that

$$
\begin{aligned}
&|Q_1^\pi(s,a) - \tilde{Q}_2^\pi(s,a)| \\
&= \left| \left(R(s,a) + \gamma \int_{s' \in S} p_1(s'|s,a) V_1^\pi(s') ds' \right) - \left(R(s,a) + \gamma \int_{s' \in S} Lp_2(s'|s,a) \tilde{V}_2^\pi(s') ds' \right) \right| \\
&\leq \gamma \left| \int_{s' \in S} p_1(s'|s,a) V_1^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') ds' \right| \\
&\leq \gamma \left| \int_{s' \in S} p_1(s'|s,a) V_1^\pi(s') - p_1(s'|s,a) \tilde{V}_2^\pi(s') + p_1(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') ds' \right| \\
&\leq \gamma \left| \int_{s' \in S} \left[p_1(s'|s,a)(V_1^\pi(s') - \tilde{V}_2^\pi(s')) + p_1(s'|s,a) \tilde{V}_2^\pi(s') - p_2(s'|s,a) \tilde{V}_2^\pi(s') \right. \right. \\
&\quad \left. \left. + p_2(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') \right] ds' \right| \\
&\leq \gamma \left| \int_{s' \in S} p_1(s'|s,a)(V_1^\pi(s') - \tilde{V}_2^\pi(s')) ds' \right| + \gamma \left| \int_{s' \in S} (p_1(s'|s,a) - p_2(s'|s,a)) \tilde{V}_2^\pi(s') ds' \right| \\
&\quad + \gamma \left| \int_{s' \in S} p_2(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') ds' \right|
\end{aligned}
$$

where the final expression was obtained by repeatedly adding and subtracting identical terms and using the triangle inequality. This expression must hold for all states s and actions a, so it must also hold for the maximum error over all states and actions:

$$
\begin{aligned}
\max_s \max_a |Q_1^\pi(s,a) - \tilde{Q}_2^\pi(s,a)| &\leq \gamma \int_{s' \in S} p_1(s'|s,a) \Delta_Q ds' + \gamma \left| \int_{s' \in S} (p_1(s'|s,a) - p_2(s'|s,a)) \tilde{V}_2^\pi(s') ds' \right| \\
&\quad + \gamma \left| \int_{s' \in S} \left(p_2(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') \right) ds' \right| \\
\Delta_Q &\leq \gamma \Delta_Q + \gamma \left| \int_{s' \in S} (p_1(s'|s,a) - p_2(s'|s,a)) \tilde{V}_2^\pi(s') ds' \right| \\
&\quad + \gamma \left| \int_{s' \in S} \left(p_2(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') \right) ds' \right| \\
&\leq \gamma \Delta_Q + \gamma V_{max} \left| \int_{s' \in S} p_1(s'|s,a) - p_2(s'|s,a) ds' \right| \\
&\quad + \gamma \left| \int_{s' \in S} \left(p_2(s'|s,a) \tilde{V}_2^\pi(s') - Lp_2(s'|s,a) \tilde{V}_2^\pi(s') \right) ds' \right| \\
&\leq \gamma \Delta_Q + \gamma V_{max} d_{var}(p_1(s'|s,a), p_2(s'|s,a)) + \gamma \varepsilon_{plan}
\end{aligned}
$$

where we have again used the triangle inequality. Therefore

$$\begin{aligned}\Delta_Q &\leq \gamma\Delta_Q + \gamma V_{\max} d_{var} + \gamma\varepsilon_{plan} \\ &= \frac{\gamma\frac{d_{var}}{1-\gamma}}{1-\gamma} + \frac{\gamma\varepsilon_{plan}}{1-\gamma},\end{aligned}$$

where we have used d_{var} as shorthand for $d_{var}(p_1(s'|s,a), p_2(s'|s,a))$

We have now expressed the error in the value function as the sum of the error due to the model approximation and the error due to using an approximate MDP planner. Using the assumptions in the lemma, the result immediately follows. ■

We can now use the prior lemmas to prove Theorem 1. First we need to examine under what conditions the two assumptions of the Simulation Lemma hold. The first assumption requires that $d_{var}(p_1(\cdot|s,a), p_2(\cdot|s,a)) \leq (1-\gamma)^2\varepsilon/(2\gamma)$ for all state-action tuples. From Lemma 6 this holds for a particular type-action tuple (which encompasses all state-action tuples where the state belongs to that type) if

$$\frac{||\beta_2 - \beta_1||_2}{\sqrt{\lambda_N}} + \sqrt{\frac{N^2\varepsilon}{\lambda_N} + \frac{2N^3 B_\sigma \varepsilon}{\lambda_N^2 - (N)^{1.5}\max_{ij}|\tilde{\sigma}_{ij} - \sigma_{ij}|\lambda_N}} \leq \frac{(1-\gamma)^2\varepsilon}{2\gamma} \tag{5}$$

and

$$\max_{ij}|\tilde{\sigma}_{ij} - \sigma_{ij}| \leq \varepsilon. \tag{6}$$

We can ensure Equation 5 holds by splitting the error into three terms:

$$\begin{aligned}\frac{\|\beta_2 - \beta_1\|_2}{\sqrt{\lambda_N}} &\leq \frac{(1-\gamma)^2\varepsilon}{4\gamma} \\ \frac{N^2\varepsilon}{\lambda_N} &\leq \frac{(1-\gamma)^4\varepsilon^2}{32\gamma^2} \\ \frac{2N^3 B_\sigma\varepsilon}{\lambda_N^2 - (N)^{1.5}\max_{ij}|\tilde{\sigma}_{ij} - \sigma_{ij}|\lambda_N} &\leq \frac{(1-\gamma)^4\varepsilon^2}{32\gamma^2}.\end{aligned}$$

Given these three equations, and Equation 6, we can obtain bounds on the error in the dynamics parameter estimates:

$$\|\tilde{\beta} - \beta\|_2 \leq \frac{(1-\gamma)^2\varepsilon\lambda_N^{0.5}}{4\gamma} \tag{7}$$

$$\max_{ij}|\tilde{\sigma}_{ij} - \sigma_{ij}| \leq \frac{(1-\gamma)^4\varepsilon^2\lambda_N}{32\gamma^2 N^2} \tag{8}$$

$$\max_{ij}|\tilde{\sigma}_{ij} - \sigma_{ij}| \leq \frac{(1-\gamma)^4\varepsilon^2\lambda_N^2}{16\gamma^2 N^3 B_\sigma + (1-\gamma)^4\varepsilon^2(N)^{1.5}\lambda_N}. \tag{9}$$

Assume[4] that $\lambda_N \leq 1, N > 1$ and $B_\sigma \geq 1$. In this case the upper bound in Equation 9 will be at least as small as the upper bounds in Equations 7 and 8.

4. This is just a simplifying assumption and it is trivial to show the bounds will have a similar polynomial dependence on the parameters if the assumptions do not hold.

We therefore require that the error ε in the model parameters be bounded by

$$\varepsilon \leq \frac{(1-\gamma)^4\varepsilon^2\lambda_N^2}{16\gamma^2N^3B_\sigma + (1-\gamma)^4\varepsilon^2(N)^{1.5}\lambda_N} \tag{10}$$

(from Equation 9). Lemma 2 provides a guarantee on the number of samples $T = \frac{12N^2B_\sigma^2}{\varepsilon^2 g}$ required to ensure with probability at least $1-g$ that all the model parameters have error of most ε. In order to ensure that the model parameters for all actions and types simultaneously fulfill this criteria with probability δ, it is sufficient to require that $g = \delta/(|A|N_T)$, from the union bound. We can then substitute this expression for g and Equation 10 into the expression for the number of samples T:

$$T = \frac{12N^2B_\sigma^2}{\left(\frac{(1-\gamma)^4\varepsilon^2\lambda_N^2}{16\gamma^2N^3B_\sigma+(1-\gamma)^4\varepsilon^2(N)^{1.5}\lambda_N}\right)\frac{\delta}{|A|N_T}} = \frac{12N^2|A|N_TB_\sigma^2(16\gamma^2N^3B_\sigma + N^1.5\lambda_N(1-\gamma)^4\varepsilon^2)^2}{(1-\gamma)^8\varepsilon^2\delta\lambda_N^4}.$$

Given this analysis, the first assumption of the Simulation Lemma holds with probability at least $1-\delta$ after

$$O\left(\frac{N^8|A|N_TB_\sigma^4}{(1-\gamma)^8\varepsilon^4\delta(\lambda_N)^4}\right)$$

samples.

The second assumption in the Simulation Lemma requires that we have access to an MDP planner than can produce an approximate solution to our typed-offset-dynamics continuous-state MDP. At least one such planner exists if the reward model is Lipschitz continuous; under a set of four conditions, Chow and Tsitsiklis (1991) proved that the optimal value function V_ε of a discrete-state MDP formed by discretizing a continuous-state MDP into $\Theta(\varepsilon)$-length (per dimension)[5] grid cells is an ε-close approximation of the optimal continuous-state MDP value function, denoted by V^*:

$$||V_\varepsilon - V^*||_\infty \leq \varepsilon.$$

The first condition used to prove the above result is that the reward function is Lipschitz-continuous. In our work, the reward function is assumed to be given, so this condition is a prior condition on the problem specification. The second condition is that the transition function is piecewise Lipschitz continuous. In other words, the transition model is Lipschitz-continuous over each of a set of finite subsets that cover the state space, and that the boundary between each subset region is piecewise smooth. For each type and action our transition model is a Gaussian distribution, which is Lipschitz-continuous, and there are a finite number of different types so it is piecewise Lipschitz-continuous. As long as our domain fulfills our earlier stated assumption that there are a finite number of different type regions, and the boundaries between each are piecewise smooth, then Chow and Tsitsiklis's second assumption is satisfied. The third condition is that the dynamics probabilities represent a true probability measure that sums to one ($\int_{s'} p(s'|s,a) = 1$), though the authors show that this assumption can be relaxed to $\int_{s'} p(s'|s,a) \leq 1$ and the main results still hold.

5. More specifically, the grid spacing h_g must satisfy $h_g \leq \frac{(1-\gamma)^2\varepsilon}{K_1+2KK_2}$ and $h_g \leq \frac{1}{2K}$ where K is the larger of the Lipschitz constants arising from the assumptions discussed in the text, and K_1 and K_2 are constants discussed in Chow and Tsitsiklis (1991). For small ε any h_g satisfying the first condition will automatically satisfy the second condition.

In our work, our dynamics models are defined to be true probability models. Chow and Tsitsiklis's final condition is that there must be a bounded difference between any two controls. In our case we consider only finite controls, so this property holds directly. Assuming the reward model fulfills the first condition, our framework satisfies all four conditions made by Chow and Tsitsiklis, and we can use their result.

By selecting fixed grid points at a regular spacing of $\frac{(1-\gamma)^2\varepsilon}{2\gamma}$, and by requiring that there exist at least one grid point placed in each contiguous single-type region, we can ensure that the maximum error in the approximate value function compared to the exactly solved value function is at most $\frac{(1-\gamma)^2\varepsilon}{2\gamma}$. This provides a mechanism for ensuring the second assumption of the Simulation Lemma holds. In other words, if the grid width used is the minimum of $\frac{(1-\gamma)^2\varepsilon}{2\gamma}$ and the minimum contiguous length of a single-type region, then the resulting value function using this discrete approximate planner is no more than $\frac{(1-\gamma)^2\varepsilon}{2\gamma}$-worse in value than the optimal exact planner for the continuous-state MDP.[6]

Type-action tuples with at least T samples are defined to be "known." From the analysis above, the estimated dynamics model for such types have a d_{var} value from the true known type-action dynamics model of at most $(1-\gamma)^2\varepsilon/(2\gamma)$. All unknown type-action tuples are defined to be self-loops. Therefore the dynamics models of our known-type, estimated dynamics MDP $\hat{M}_K$ relative to a known-type MDP with the true dynamics parameters M_K have a d_{var} of zero for all the unknown type-action tuples (since these are always defined as self loops) and at most $(1-\gamma)^2\varepsilon/(2\gamma)$ for all the known type-action tuples. Hence the first assumption of the Simulation Lemma holds. The second assumption of the Simulation Lemma is fulfilled given the analysis in the prior paragraph. Given these two assumptions are satisfied, the Simulation Lemma guarantees that the approximate value of our known-type MDP $\hat{M}_K$ under its greedy policy π ($\pi(s) = \text{argmax}_a Q_{\hat{M}_K}(s,a)$) is ε-close to the optimal value of the known-type MDP with the true dynamics parameters M_K under policy π: $||\tilde{V}^{\pi}_{\hat{M}_K} - V^{\pi}_{M_K}||_\infty \leq \varepsilon$. This fulfills condition 2 of Strehl et al. (2006).

The first condition of Strehl et al. (2006) can be re-expressed as:

$$Q^*(s,a) - Q_{\hat{M}_K}(s,a) = (Q^*(s,a) - Q_{M_K}(s,a)) + (Q_{M_K}(s,a) - Q_{\hat{M}_K}(s,a)) \leq \varepsilon.$$

We start by considering the first expression, $Q^*(s,a) - Q_{M_K}(s,a)$. If all type-action pairs are known, then M_K is the same as the original MDP, and this expression equals 0. If some type-action pairs are unknown, then the value of states of that type, associated with that action, becomes $V_{\max}$ under MDP M_K. As all known type-action pairs have the same reward and dynamics model as the original MDP, this implies that the value Q_{M_K} must be either equal or greater than Q^*, since all the value of all unknown state-actions is at least as great in Q_{M_K} as their real value Q^*. For this reason, $Q^*(s,a) - Q_{M_K}(s,a)$ is always less than or equal to 0.

We next consider $Q_{M_K}(s,a) - Q_{\hat{M}_K}(s,a)$. The variational distanced_{var} between the dynamics models of M_K and $\hat{M}_K$ for all unknown type-action tuples is zero, because all the dynamics of unknown tuples are self loops. As discussed above, the d_{var} between all known type-action tuples is at most $(1-\gamma)^2\varepsilon/(2\gamma)$. We can then apply the Simulation Lemma to guarantee that $|Q^{\pi}_{M_K}(s,a) - Q^{\pi}_{\hat{M}_K}(s,a)| \leq \varepsilon$. As a result, the first condition of Strehl et al. (2006) holds.

The third condition limits the number of times the algorithm may experience an unknown type-action tuple. Since there are a finite number of types and actions, this quantity is bounded above by

6. The condition of the extent of a typed region is a requirement in order to ensure that the discrete-representation doesn't skip over a smaller region of a different type, that may have a different optimal policy.

$N_{at}N_T|A|$, which is a polynomial in the problem parameters $(N, |A|, N_T, \frac{1}{\varepsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma}, \frac{1}{\lambda_N}, B_\sigma)$. Therefore, our algorithm fulfills the three specified criteria and the result follows. ■

3.3 Discussion

Prior PAC-MDP work has focused predominantly on discrete-state, discrete-action environments. The sample complexity of the R-max algorithm by Brafman and Tennenholtz (2002) scales with the number of actions and the square of the number of discrete states, since a different dynamics model is learned for each discrete state-action tuple. In environments in which states of the same type share the same dynamics, Leffler et al. (2007) proved that the sample complexity scales with the number of actions, number of discrete states, and number of types. Assuming the number of types is typically much less than the number of states, this can result in significantly faster learning, as Leffler et al. (2007) demonstrate empirically. However, a naïve application of either technique to a continuous-state domain involves uniformly discretizing the continuous-state space. This procedure that results in a number of states that grows exponentially with the dimension of the state space. In this scenario the approaches of both Brafman and Tennenholtz (2002) and Leffler et al. (2007) will have a sample complexity that scales exponentially with the state space dimension, though the approach of Leffler et al. (2007) will scale better if there are a small number of types.

In contrast, the sample complexity of our approach scales polynomially in the numbers of actions and types as well as state space dimension, suggesting that it is more suitable for high dimensional environments. Our results follow the results of Strehl and Littman (2008), who gave an algorithm for learning in continuous-state and continuous-action domains that has a sample complexity that is polynomial in the state space dimension and the action space dimension. Our work demonstrates that we can get similar bounds when we use a more powerful dynamics representation (allowing states to have different dynamics, but sharing dynamics within the same types), learn from experience the variance of the dynamics models, and incorporate the error due to approximate planning.

Our analysis presented so far considers the discounted, infinite-horizon learning setting. However, our results can be extended to the H-step finite horizon case fairly directly using the results of Kakade (2003). Briefly, Kakade considers the scenario where a learning algorithm $\mathcal{A}$ is evaluated in a cycling H-step periods. The H-step normalized undiscounted value U of an algorithm is defined to be the sum of the rewards received during a particular H-step cycle, divided by H. Kakade defines $\mathcal{A}$ to be ε-optimal if the value over a state-action trajectory, until the end of the current H-step period, is within ε of the optimal value U. Using this alternate definition of value requires a modification of the Simulation Lemma which results in a bound on Δ_Q of $(H-1)V_{max}d_{var} + (H-1)\varepsilon_{plan}$. In short, the $\gamma/(1-\gamma)$ terms has been replaced with $H-1$. The earlier results on the number of samples required to obtain good estimates of the model parameters are unchanged, and the final result follows largely as before, except we now have a polynomial dependence on the horizon H of the undiscounted problem, compared to a polynomial dependence on the discount factor $1/\gamma$.

Finally, though our focus is on sample complexity, it is also important to briefly consider computational complexity. To ensure the approximate planner produces highly accurate results, our algorithm's worst-case computational complexity is exponential in the number of state dimensions. While this fact prevents it from being theoretically computationally efficient, in the next section we

present experimental results that demonstrate our algorithm performs well empirically compared to a related approach in a real-life robot problem.

4. Experiments

First we demonstrate the benefit of using domain knowledge about the structure of the dynamics model on the standard reinforcement-learning benchmark, PuddleWorld (Boyan and Moore, 1995). Then we illustrate the importance of explicitly learning the variance parameters of the dynamics in a simulated "catching a plane" experiment. This is in contrast to some past approaches which assume the variance parameters to be provided, such as Strehl and Littman (2008).

Our central hypothesis is that offset dynamics are a simple model that reasonably approximate several real world scenarios. In our third experiment, we applied our algorithm to a simulated problem using in which an agent needs to learn the best route to drive to work, and used date collected from real cars to simulate the dynamics. In our final experiment we applied our algorithm to a real-world robot car task in which the car most cross varying terrain (carpet and rocks) to reach a goal location.

CORL requires a planner to solve the estimated continuous-state MDP. Planning in continuous-state MDPs is an active area of research in its own right, and is known to be provably hard (Chow and Tsitsiklis, 1989). In all our experiments we used a standard technique, Fitted Value Iteration (FVI), to approximately solve the current MDP. In FVI, the value function is represented explicitly at only a fixed set of states. In our experiments these fixed states are uniformly spaced in a grid over the state space. Planning requires performing Bellman backups for each grid point; the value function over points not in this set is computed by function interpolation. We used Gaussian kernel functions as the interpolation method. Using sufficiently small kernel widths relative to the spacing of the grid points will make the approach equivalent to using a nearest neighbour standard discretization. However, there are some practical advantages in coarse grids to more smooth methods of interpolation. We discuss this issue in more depth in the next section.

In each experiment, N_{at} was tuned based on informal experimentation.

4.1 Puddle World

Puddle world is a standard continuous-state reinforcement-learning problem introduced by Boyan and Moore (1995). The domain is a two-dimensional square of width 1 with two oval puddles, which consist of the area of radius 0.1 around two line segments, one from $(0.1, 0.75)$ to $(0.45, 0.75)$ and the other from $(0.45, 0.4)$ to $(0.45, 0.8)$. The action space consists of the four cardinal directions. Upon taking an action the agent moves 0.05 in the specified cardinal direction with added Gaussian noise $\mathcal{N}(0, 0.001 * I)$, where I is a two-by-two identity matrix. The episode terminates when the agent reaches the goal region which is defined as the area in which $x + y \geq 1.9$. All actions receive a reward of -1 unless the agent is inside a puddle, in which case it then receives a reward of -400 times the distance inside the puddle. Figure 1 provides a graphical depiction of the puddle world environment.

We expect CORL will outperform prior approaches on this problem for two reasons. The first is that we assume the reward is given. However even if the reward model is provided, most past work still has to learn the dynamics model for this world, which is still a significant undertaking. The second reason is a feature of the CORL algorithm: its dynamics model uses the additional information that the dynamics for one action for all states of the same type are identical. Here there

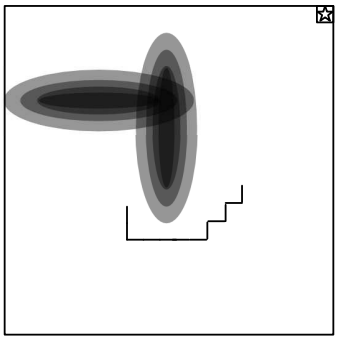


Figure 1: Puddle world. The small square with a star in it denotes the goal region. The black ovals represent the puddles and the black line represents a sample partial trajectory of an agent navigating around this world.

is only a single type, so CORL must learn only 4 sets of transition parameters, one for each action. Our goal here is simply to demonstrate that this additional information can lead to significantly faster learning over other more general techniques.

Puddle world has previously been used in reinforcement-learning competitions, and our evaluation follows the procedure of the second bakeoff (Dutech et al., 2005). We initially generated 50 starting locations, and cycled through these when initializing each episode. Each episode goes until the agent reaches the goal, or has taken 300 actions. We report results from taking the average reward within fifty sequential episodes (so the first point is the average reward of episodes 1-50, the second point is the average of episodes 51-100, etc.). We set N_{at} to be 15, and solved the approximate MDP model using Fitted Value Iteration with Gaussian kernel functions. The kernel means were spaced uniformly in a 20x20 grid across the state space (every 0.05 units), and their standard deviation was set to 0.01. Note that in the limit as the standard deviation goes to 0 the function interpolation becomes equivalent to nearest neighbour. Nearest neighbour is the interpolation method used in the approximate continuous-state MDP solver by Chow and Tsitsiklis (1991) which provides guarantees on the resulting value function approximation. However, since computational complexity scales exponentially with the grid discretization, practical application can require the use of coarse grids. In this case, we found that using a smoother function interpolation method empirically outperformed a nearest neighbour approach. In particular, we found that a kernel width, which is a measure of the standard deviation, of 0.01 gave the best empirical performance. This value lies in the middle of kernel widths which are smaller than the variance of the dynamics models and the grid spacing and those widths larger than the grid spacing and dynamics models.

We compare our results to the reported results of Jong and Stone (2007)'s Fitted R-max and to Lagoudakis and Parr (2003)'s Least Squares Policy Iteration (LSPI).[7] Fitted R-max is an instance-based approach that smoothly interpolates the dynamics of unknown states with previously observed transitions, and takes a similar approach for modeling the reward function. LSPI is a policy iteration approach which uses a linear basis function representation of the state-action values, and uses a set of sample transitions to compute the state-action values.

7. Both results reported come from the paper of Jong and Stone (2007).

Algorithm	Number of episodes			
	50	100	200	400
CORL	−19.9	−18.4	−20.3	−18.7
Fitted R-max	−130	−20	20	−20
LSPI	−500	−330	−310	−80

Table 1: PuddleWorld results. Each entry is the average reward/episode received during the prior 50 episodes. Results for the other algorithms are as reported by Jong and Stone (2007).

On the first 50 episodes, Fitted R-max had an average reward of approximately -130, though on the next 50 episodes it had learnt sufficiently good models that it reached its asymptotic performance of an average reward of approximately -20. In contrast, CORL learned a good model of the world dynamics within the very first episode, since it has the advantage of knowing that the dynamics across the entire state space are the same. This meant that CORL performed well on all subsequent episodes, leading to an average reward on the first 50 episodes of -19.9. Least Squares Policy Iteration (Lagoudakis and Parr, 2003) learned slower than Fitted R-max. Results are summarized in Table 1.

It is worth a short remark on the comparability of the reported results, as LSPI and Fitted R-max were run without knowing the reward model. LSPI's performance on a deterministic reward reinforcement-learning problem such as PuddleWorld will be identical to its performance in the known-reward case, as the rewards in the sampled transitions will be the same in either situation. Given this, assuming known reward, as we do for CORL, will not change the LSPI results. In contrast we do expect that Fitted R-max will be slightly faster if it does not have to learn the reward model. This is because a wider interpolation width can be selected if the reward is known and only the dynamics are unknown, since all states share the same dynamics. However, even in this case, Fitted R-max will be approximating the Gaussian dynamics by a set of observed transitions, and so it appears likely that CORL will still be faster than Fitted R-max since CORL assumes the (true) parametric representation of the transition model.

In summary, given the reward function, CORL can learn a good dynamics model extremely quickly in PuddleWorld, since the dynamics are typed-offset with a single type. This additional information enables CORL to learn a good policy for puddle world much faster than Fitted R-max and LSPI. This experiment illustrates the advantage of CORL when the transition dynamics are known to be identical across the state space, and to follow a noisy offset model.

4.2 Catching a Plane

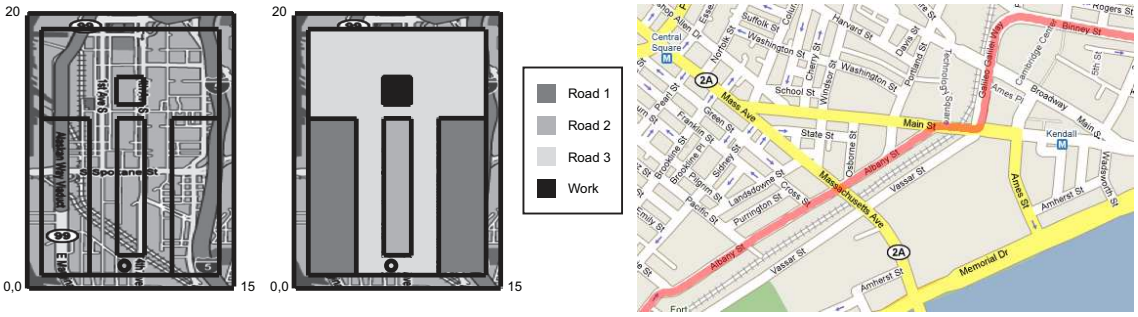
We next consider some examples with multiple types. Thousands of people now rely on internet maps or GPS units to do efficient trip planning, and better traffic prediction is an area of recent research (Horvitz et al., 2005). In many street systems, there exist a number of different classes of road types according to the designated speed limits associated with these roads. These different road types are often also associated with different variances. For example, while highways have mean speeds that are faster than small side streets, highways typically have a very large variance; rush hour highway speeds may often be slower than smaller side streets.

Figure 2: Motivating example where an agent must drive from a start location (designated with an S) to a goal area (shown with a highlighted rectangle) in time for a deadline, and can choose to take large high speed and variance roads, or slower small variance roads.

In our first experiment we considered a scenario in which learning this variance is critical: an agent must learn the best way to drive to an airport in time to catch a plane. A similar real-world environment is depicted in Figure 2. The agent starts from home and can either drive directly along a small side street, or cross over to the left or right to reach a main highway. The agent goes forward more quickly on the highway (with a mean offset of 2 units), but the highway also has a high variance of 0.49. In contrast, on the small side streets the agent goes forward more slowly (a mean offset of 1 unit) but with very small variance (0.00001). The state space is four dimensional, consisting of the agent's current x and y location, its orientation, and the amount of time that has passed. On each step five minutes pass. The agent can drive in a 14 by 19 region: outside of this is considered to be too far away.[8] If the agent exits this region it receives a reward of -1. The cost for each step is -0.05, the reward for reaching the airport in time for check in is $+1$ and the cost for not making the airport in time is -1. The discount factor is set to 1.0. N_{at} was set to 10. An episode starts when the agent takes its first step and lasts until the agent reaches the goal, exits the allowed region, or reaches the time at which check in for the plane closes. The agent starts at location $7,3$ facing north, and must figure out a way to reach the goal region which spans $[6.5-8.5, 15.5\text{-}17.5]$. To solve the underlying MDP, fitted value iteration was used. The fixed points were regularly spaced grid points with 15 across the x dimension, 20 across the y dimension, 4 orientation angles, and 21 time intervals. This yielded over 25,000 fixed points.

The correct path to take depends on the amount of time left. Here we explored three different deadline scenarios: when the agent has 60 minutes remaining (*RunningLate*), 70 minutes remaining (*JustEnough*), or 90 minutes remaining until check in closes for the plane (*RunningEarly*). In all three scenarios, the agent learned a good enough model of the dynamics within 15 episodes to compute a good policy. Note that using a naïve discretization of the space with the FVI fixed points as states and applying R-max would be completely intractable, as a different model would have to be learned for each of over 25,000 states. We display results for all three scenarios in Table 2. In

8. Note that this could be a reasonable assumption in some cases, such as when doctors had to stay within a certain radius of a hospital when on call.



(a) (b) (c)

Figure 3: (a) Simulated world map showing example environment. (b) World divided into a set of discrete types. Agent starts at driveway (black circle) and tries to learn a good route to reach work. (c) An example section of a car trajectory (constructed from a set of GPS locations).

the *RunningLate* scenario the agent learned that the side streets are too slow to allow it to ever reach the airport in time. Instead it took the higher variance highway which enables it to reach the airport in time in over half the episodes after N_{at} is reached for all types: its average reward is -0.4833 and it takes on average 56.62 minutes for it to reach the airport. In *JustEnough* the agent learned that the speed of side streets is sufficiently fast for the agent to reach the airport consistently in time, whereas the higher speed and variance highway would result in the agent failing to reach the check in time in some cases. Here the agent always reached the goal and receives an average reward of 0.45. In the *RunningEarly* scenario, the agent has enough time so that it can take either route and reliably reach the airport in time. In this scenario it learned to always take the highway, since in expectation that route will be faster. The average reward here was 0.4629.

This simulation serves to illustrate that our algorithm can quickly learn to perform well, in situations in which learning the variance is critical to ensure good performance.

4.3 Driving to Work

In our second experiment we again consider a simulated trip routing problem, but we now generate transitions in the simulator by sampling from real traffic data distributions. Here an agent must learn the best series of actions to drive from home to work in a small simulated world (see Figure 3(a) and 3(b)). The state consists of the current coordinates (x, y) and the orientation of the agent. There are three road types and each road type is associated with a different distribution of speeds. The

Scenario	Deadline (min)	Mean Reward/Episode	Mean Time to Reach Goal (min)
RunningLate	60	-0.4833	56.62
JustEnough	70	0.45	60
RunningEarly	90	0.4629	58.6

Table 2: Catching a plane: results after N_{at} has been reached for all types.

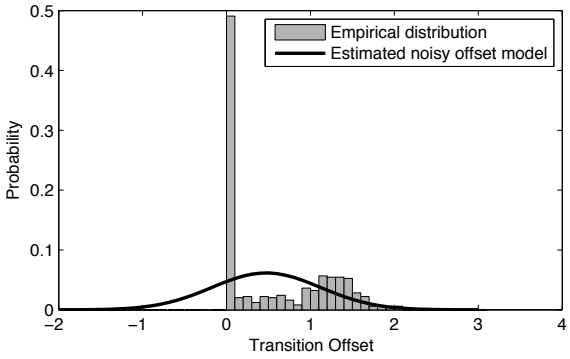
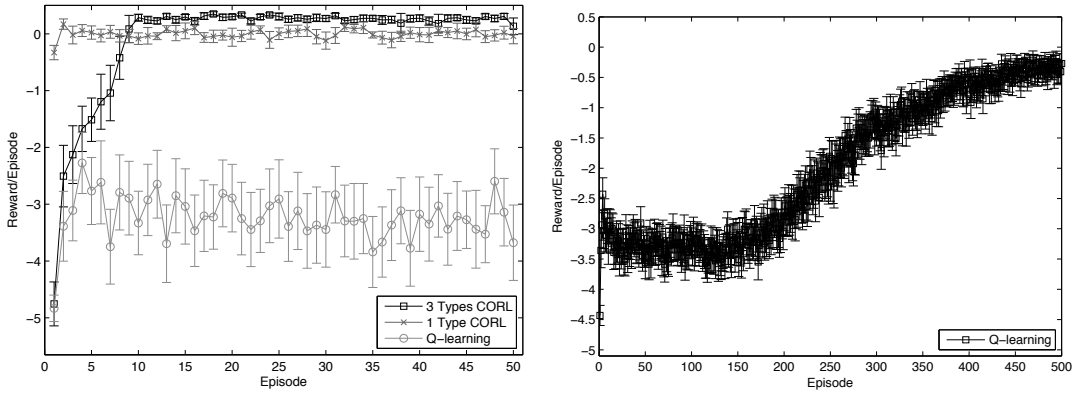


Figure 4: A histogram of car speeds on small roads that is used to generate transitions on road type 2 and the estimated dynamics model parameters found during the experiment

distributions were obtained from the CarTel project (Eriksson et al., 2008), which consists of a set of car trajectories from the Boston, Massachusetts area. GPS locations and time stamps are stored approximately every second from a fleet of 27 cars.[9] A section from one car trajectory is shown in Figure 3(c). Using this data set we extracted car trajectories on an interstate highway, small side streets and a local highway: these constitute types 1, 2 and 3 respectively in the simulation world. Each car trajectory consisted of a set of D GPS+time data points, which was converted into a set of $D-1$ transitions. Each transition in the simulation was sampled from these real-world transitions; for example, transitions in the simulator on road type 2 were sampled from real-world transitions on small side streets. Transitions from all three road types were all rescaled by the same constant in order to make the distances reasonable for the simulated world.[10] Figure 4 displays a histogram of rescaled transitions associated with small side streets. This figure shows that the speed distribution for small side streets was not Gaussian: the speed distribution for the other two street types was also not Gaussian. In particular, in no trajectories used does the car ever go backwards, whereas in some Gaussian models there will be small probability of this occurring. In this experiment we sought to investigate how well a noisy offset model could function in this environment, and the benefit of directly modelling different types of roads. Each transition in the simulated environment was sampled from the histogram of speeds associated with the road type at the agent's current position. Therefore, the data from the simulator is closer to the real environment than to the Gaussian distributions assumed by the learning algorithm.

The agent received a reward of 1 for reaching the work parking lot, -0.05 for each step, and -1 if it left the local area. Each episode finished when the agent either reached the goal, left the local area, or had taken 100 steps. An agent can go left, right or straight at each step. The transition induced by a straight action was determined by the road type as specified in the prior paragraph, and going left or right changed the orientation of the agent by 90 degrees with a very small amount of noise. The number of samples needed until a type-action tuple is known, N_{at}, was set to be 20. The discount factor was 1. The agent was always started in the same location and was allowed to learn across a set of 50 episodes. Results were averaged across 20 rounds of 50 episodes per

9. See more information about the project at `http://cartel.csail.mit.edu/`.
10. We also removed outlier transitions, as extremely fast speeds/transitions were likely to be errors in the log file.



(a) (b)

Figure 5: Reward versus episode. (a) Compares CORL with 3 types and 1 type to Q-learning. Results are averaged over 20 rounds (50 episodes per round). Error bars show 95% confidence intervals. (b) Shows Q-learning with 500 episodes per round, averaged over 100 rounds.

round. In one experiment the agent was given full knowledge of the three world types, and learned a different dynamics model for each type and action. In the second experiment the agent assumed there was only a single type and learned a dynamics model for each action. We also compared our approach to *Q*-learning over a uniformly-spaced discrete grid over the environment with an ε-greedy policy. We used a discretization that was identical to the fixed points used in the fitted value iteration planner of CORL. Points were mapped to their nearest neighbors. Q-learning requires specifying two parameters: the learning rate α which determines how much to adjust the state-action value estimates after each update, and ε which specifies how often to take a random action instead of the action that maximizes the current Q values. In this experiment α was set to 1.0 and decreased by multiplying by a factor of 0.9999 at each step.[11] We set ε to be 0.1.

The CORL results are displayed in Figure 5(a). This figure displays three encouraging results. The first is that in both CORL algorithms the agent learned to consistently reach the goal: the only way that the agent can receive a reward greater than -1 is to reach the goal, and all confidence intervals lie above -1 for all episodes after 10, indicating that the agent in both cases was successfully reaching the goal. This is promising because even though the underlying dynamics models were not exactly Gaussian noisy offset dynamics, a noisy offset model approximation was sufficient for the agent to learn a good policy in this environment. The estimated parameters computed for one type and action are displayed in Figure 4.

The second result is that the policy found by the agent that models all three types differently resulted in significantly higher reward than modeling the world with a single type: its performance suffered initially because it takes longer to learn a model of the world dynamics, but from about episode 10-50 modelling all types separately resulted in significantly higher reward per episode than modelling all types as the same. Table 3 displays the average reward of both approaches on episodes 10-50. These results demonstrate that traffic data does exhibit different speed distributions

11. We tried different decay factors for the α parameter but found that this worked better than decaying α more rapidly.

Algorithm	Average reward/episode
CORL with 3 types	0.27
CORL with 1 type	0.00
Q-learning	−3.2485

Table 3: Average reward on episodes 10-50 for the driving to work example.

on different types of roads, and that by considering such differences CORL can improve route directions even in a small simulated example.

The third result is that both CORL algorithms significantly outperformed Q-learning: again see Table 3 for comparing the short term performance of Q-learning to the CORL algorithm. This is not surprising since Q-learning is a model-free approach that trades off speed of computation per step in return for not requiring consistency between its state values through learned reward and dynamics models. Here in particular there is a large amount of structure in the domain that Q-learning cannot use. Q-learning does eventually begin to consistently reach the goal but this is only after about 500 episodes, more than an order of magnitude longer than the CORL algorithms took to find a good policy. These results are displayed in Figure 5(b). Such results argue that in situations where data is costly to gather, using a model can be extremely helpful.

4.4 Robot Navigation Over Varying Terrain

We also tried our algorithm in a real-life robotic environment involving a navigation task where a robotic car must traverse multiple surface types to reach a goal location. This experiment is a second example where a noisy offset dynamics model provides a sufficiently good representation of the real-world dynamics to allow our algorithm to learn good policies. We compared to the RAM-Rmax algorithm (Leffler et al., 2007), a provably efficient RL algorithm for learning in discrete-state worlds with types. The authors demonstrated that, by explicitly representing the types, they could get a significant learning speedup compared to R-max, which learns a separate dynamics model for each state. The RAM-Rmax algorithm represents the dynamics model using a list of possible next outcomes for a given type. CORL works directly with continuous-valued states, resulting in the improved sample complexity discussed earlier. This is achieved through assuming a fixed parametric representation of the dynamics, which is a less flexible model than the one used in RAM-Rmax. In this experiment we were interested in whether our representation was still rich enough to capture the real world dynamics involved in varying terrain traversal. We also investigated the computational load of CORL compared to RAM-Rmax, since by restricting our representation size we hoped to also achieve computational savings.

In this experiment we ran a LEGO® Mindstorms NXT robot (see Figure 6(b)) on a multi-surface environment. A tracking pattern was placed on the top of the robot and an overhead camera was used to determine the robot's current position and orientation. The domain, shown in Figure 6(a), consisted of two types of terrain: rocks embedded in wax and a carpeted area. The goal was for the agent to begin in the start location (indicated in the figure by an arrow) and end in the goal without going outside the environmental boundaries. The rewards were -1 for going out of

(a) (b)

Figure 6: (a) Image of the environment. The start location and orientation is marked with an arrow. The goal location is indicated by the circle.(b) LEGO® robot.

bounds, $+1$ for reaching the goal, and -0.01 for taking an action. Reaching the goal and going out of bounds ended the episode and resulted in the agent getting moved back to the start location.[12]

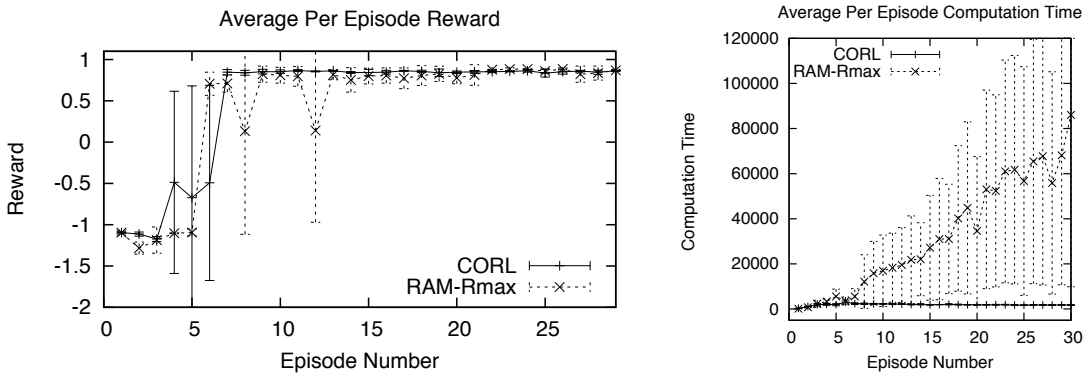
Due to the close proximity of the goal to the boundary, the agent needs an accurate dynamics model to reliably reach the goal. Part of the difficultly of this task is that the actions were going forward, turning left, and turning right. Without the ability to move backwards, the robot needed to approach the goal accurately to avoid falling out of bounds.

For the experiments, we compared our algorithm ("CORL") and the RAM-Rmax algorithm ("RAM"). The fixed points for the fitted value iteration portion of our algorithm were set to the discretized points of the RAM-Rmax algorithm. Both algorithms used an EDISON image segmentation system to uniquely identify the current surface type. The reward function was provided to both algorithms.

The state space is three dimensional: x, y position and orientation. Our algorithm implementation for this domain used a full covariance matrix to model the dynamics variance. For the RAM-Rmax agent, the world was discretized to a forty-by-thirty-by-ten state space. In our algorithm, we used a function approximator of a weighted sum of Gaussians, as described in Section 2. We used the same number of Gaussians to represent the value function as the size of the state space used in the discretized algorithm, and placed these fixed Gaussians at the same locations. The variance over the x and y variables was independent of each other and of orientation, and was set to be 16. To average orientation vectors correctly (so that -180° degrees and 180° do not average to 0) we converted orientations θ to a Cartesian coordinate representation $x_\theta = \cos(\theta), y_\theta = \sin(\theta)$. The variance over these two was set to be 9 for each variable (with zero covariance). For our algorithm and the RAM-Rmax algorithm, the value of N_{at} was set to four and five, respectively, which was determined after informal experimentation. The discount factor was set to 1.

Figure 7(a) shows the average reward with standard deviation for each of the algorithms over three runs. Both algorithms are able to receive near-optimal reward on a consistent basis, choosing

12. A video of the task can be seen at `http://people.csail.mit.edu/emma/corl/SuccessfulRun.mov` and `http://people.csail.mit.edu/emma/corl/SuccessfulRun.wmv`.



(a) (b)

Figure 7: (a) Reward received by algorithms averaged over three runs. Error bars show one standard deviation. (b) Total time taken by algorithms averaged over three runs. Error bars show one standard deviation.

similar paths to the goal. Our dynamics representation is sufficient to allow our algorithm to learn well in this real-life environment.

In addition, by using a fixed size (parametric) dynamics representation, the computational time per episode of our algorithm is roughly constant (Figure 7(b)). In the implementation of RAM-Rmax, the computational time grew with the number of episodes due to its dynamics model representation. This suggests that using a fixed size dynamics representation can have significant computation benefits. Overall CORL performed well in this domain, both in terms of reward achieved and computation required.

5. Conclusion and Future Work

In this paper we have presented CORL, an algorithm for efficiently learning to act in typed, continuous-state environments. CORL has a sample complexity that scales polynomially with the state space dimension and the number of types: this bound also directly incorporates the error due to approximate planning. Experiments on a simulated driving example using real world car data, and a small robot navigation task, suggest that noisy offset dynamics are a sufficiently rich representation to allow CORL to perform well in some real-world environments.

Due to the approximate MDP planning, we cannot currently guarantee both polynomial sample complexity and polynomial computational complexity. There are a number of recent advances in continuous-state MDP planning (Kocsis and Szepesvári, 2006; Kveton and Hauskrecht, 2006; Marecki and Tambe, 2008) as well as alternate approaches such as forward search techniques. In the future it would be interesting to investigate whether there exist alternate MDP planners that can provide ε-close approximations to the exact solutions with a computational complexity that scales polynomially with the number of state dimensions. Such approaches would enable CORL to achieve the appealing goal of polynomial dependence on the number of state dimension for both sample complexity and computational complexity.

Finally, the bounds provided remain overly large for many practical applications. We are broadly interested in developing techniques that can tighten the gap between the theoretical bounds and those needed for practical performance in real-world reinforcement learning.

Acknowledgments

B. Leffler, L. Li and M. Littman were partially supported by NSF DGE 0549115, a DARPA Transfer Learning grant and a National Science Foundation (NSF) Division of Information and Intelligent Systems (IIS) grant. E. Brunskill and N. Roy were supported by NSF IIS under Grant #0546467. Special thanks to Jacob Eriksson, Hari Balakrishnan, Samuel Madden, and the rest of the MIT CarTel team for generously providing access to their data set. We also appreciate the gracious time of the reviewers in helping us improve this work.

Appendix A.

Lemma 5 *Assume* $\max_d |\tilde{\beta}_d - \beta_d| \leq \varepsilon$ *for* $\varepsilon < 1/4$. *Given any* $\delta > 0$, *define* $T_\sigma = \frac{12N^2B_\sigma^2}{\delta\varepsilon^2}$. *If there are* T_σ *transition samples* (s,a,s'), *then with probability at most* $\frac{\delta}{3}$, *the estimated covariance parameter* $\tilde{\sigma}_{ij}$, *computed by Equation 3, deviates from the true covariance parameter* σ_{ij} *by more than* ε *over all entries* ij; *formally,* $\Pr(\max_i |\tilde{\sigma}_{ij} - \sigma_{ij}| \geq \varepsilon) \leq \frac{\delta}{3}$.

Proof First recall that σ_{ij} represents the covariance between dimensions i and j. We are interested in the probability that the estimated covariance $\tilde{\sigma}_{ij}$ differs from the true parameter σ_{ij}: $\Pr(|\tilde{\sigma_{ij}} - \sigma_{ij}| \geq \varepsilon)$. From Chebyshev's inequality, we can bound this expression as

$$\Pr(|\tilde{\sigma_{ij}} - \sigma_{ij}| \geq \varepsilon) \quad \leq \quad \frac{Var(\tilde{\sigma}_{ij})}{\varepsilon^2}, \tag{11}$$

where $Var(\tilde{\sigma}_{ij})$ is the variance of the sample variance.

We therefore require an upper bound on the variance of the sample covariance. We will derive a bound on this below in the general case of the covariance between two variables x and y both of which are Gaussian distributed.

$$\begin{aligned} Var(\tilde{\sigma}_{xy}) &= E[(\tilde{\sigma}_{xy} - \sigma_{xy})^2] \\ &= E\left[\left(\frac{1}{T_\sigma}\sum_{k=1}^{T_\sigma}(x_k - \bar{x})(y_k - \bar{y}) - \sigma_{xy}\right)^2\right] \end{aligned}$$

where $\bar{x}$ and $\bar{y}$ are the respective sample means, and in the second line we have written out the definition of the sample covariance. We can then use the linearity of expectation to derive

$$\begin{aligned} Var(\tilde{\sigma}_{xy}) &= \frac{1}{T_\sigma^2}\sum_{k=1}^{T_\sigma}\sum_{m=1}^{T_\sigma} E[(x_k - \bar{x})(x_m - \bar{x})(y_k - \bar{y})(y_m - \bar{y})] \\ &\quad -2\sigma_{xy}\frac{1}{T_\sigma}\sum_{k=1}^{T_\sigma} E[(x_m - \bar{x})(y_m - \bar{y})] + E[(\sigma_{xy})^2] \\ &= \frac{1}{T_\sigma^2}\sum_{k=1}^{T_\sigma}\sum_{m=1}^{T_\sigma} E[(x_k - \bar{x})(x_m - \bar{x})(y_k - \bar{y})(y_m - \bar{y})] - (\sigma_{xy})^2 \end{aligned}$$

where the second line follows from the definition of the covariance σ_{xy}. We next divide the summation into two expressions, when $m = k$ and when $m \neq k$, and use the property that the expectation of independent variables is the product of their expectations:

$$\begin{aligned} Var(\tilde{\sigma}_{xy}) &= \frac{1}{T_\sigma} E[(x_k - \bar{x})^2 (y_k - \bar{y})^2] + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} E[(x_k - \bar{x})(y_k - \mu_k)] E[(x_m - \bar{x})(y_m - \bar{y})] - (\sigma_{xy})^2 \\ &= \frac{1}{T_\sigma} E[(x_k - \bar{x})^2 (y_k - \bar{y})^2] + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} (\sigma_{xy})^2 - (\sigma_{xy})^2. \end{aligned}$$

We can now use the Cauchy-Schwarz inequality on the first term to get

$$\begin{aligned} Var(\tilde{\sigma}_{xy}) &\leq \frac{1}{T_\sigma} \sqrt{E[(x_k - \bar{x})^4] E[(y_k - \bar{y})^4]} + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} (\sigma_{xy})^2 - (\sigma_{xy})^2 \\ &= \frac{1}{T_\sigma} \sqrt{E[(x_k + \mu_x - \mu_x - \bar{x})^4] E[(y_k + \mu_y - \mu_y - \bar{y})^4]} + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} (\sigma_{xy}^2)^2 - (\sigma_{xy}^2)^2 \\ &= \frac{1}{T_\sigma} \sqrt{(3\sigma_{xx}^4 + 6\sigma_{xx}^2 (\bar{x} - \mu_x)^2 + (\bar{x} - \mu_x)^4)(3\sigma_{yy}^4 + 6\sigma_{yy}^2 (\bar{y} - \mu_y)^2 + (\bar{y} - \mu_y)^4)} \\ &\quad + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} (\sigma_{xy})^2 - (\sigma_{xy})^2 \end{aligned}$$

where we have used the fact that the fourth central moment of a Gaussian distribution is $3\sigma_{xx}^2$ in the final line. Next we make use of the assumptions that B_σ is an upper bound to all covariance matrix elements and the bound on the maximum error in the parameter offset estimates:

$$\begin{aligned} Var(\tilde{\sigma}_{xy}^2) &\leq \frac{(\varepsilon^4 + 6\varepsilon^2 B_\sigma + 3B_\sigma^2)}{T_\sigma} + \frac{T_\sigma (T_\sigma - 1)}{T_\sigma^2} (\sigma_{xy})^2 - (\sigma_{xy})^2 \\ &\leq \frac{4B_\sigma^2}{T_\sigma} \end{aligned}$$

where the last line follows because $\varepsilon < 1/4$ and $B_\sigma \geq 1$. We can then substitute this result into Equation 11 which yields

$$P(|\tilde{\sigma}_{ij} - \sigma_{ij}| \geq \varepsilon) \leq \frac{4B_\sigma^2}{\varepsilon^2 T_\sigma}.$$

To ensure that this bound holds simultaneously with probability $\frac{\delta}{3}$ for all N^2 covariance matrix elements it suffices by the union bound to require that each covariance entry exceeds its expected value by more than ε with probability at most $\frac{\delta}{3N^2}$:

$$\frac{4B_\sigma^2}{\varepsilon^2 T_\sigma} \leq \frac{\delta}{3N^2}.$$

Re-arranging yields the bound for the required number of samples:

$$T_\sigma \geq \frac{12N^2 B_\sigma^2}{\delta \varepsilon^2}.$$

■

Lemma 8 *If* $\max_{i,j}|\Sigma_1(i,j)-\Sigma_2(i,j)| \le \varepsilon$ *for any* $1 \le i,j \le N$, *then*

$$\left|\ln\frac{\det\Sigma_2}{\det\Sigma_1}\right| \le N\varepsilon\left(\frac{1}{\lambda_1}+\frac{1}{\lambda_2}+\cdots+\frac{1}{\lambda_N}\right) \le \frac{N^2\varepsilon}{\lambda_N}.$$

Proof Define $E=\Sigma_2-\Sigma_1$. Clearly, E is symmetric since both Σ_1 and Σ_2 are symmetric. Its eigenvalues are denoted by $\psi_1 \ge \psi_2 \ge \cdots \ge \psi_N$, which are real (but can be negative or positive). First, it is known that

$$\det\Sigma_1=\prod_{i=1}^{N}\lambda_i \quad \& \quad \det\Sigma_2=\prod_{i=1}^{N}\lambda_i'.$$

Therefore,

$$\ln\frac{\det\Sigma_2}{\det\Sigma_1}=\ln\prod_{i=1}^{N}\frac{\lambda_i'}{\lambda_i}=\sum_{i=1}^{N}\ln\frac{\lambda_i'}{\lambda_i}.$$

From Geršgorin's theorem (Horn and Johnson, 1986, Theorem 6.1.1), the eigenvalues of E must be small as the elements of E are small. Specifically, each ψ_i must lie in one of the n Geršgorin discs:

$$\forall 1 \le j \le N : D_j=\{x \in R \mid |x-E(j,j)| \le \sum_{j' \ne j}\left|E(j,j')\right|\}.$$

It follows immediately that

$$|\psi_i| \le \sum_{j=1}^{N}|E(i,j)| \le N\varepsilon$$

as every component in E lies in $[-\varepsilon,\varepsilon]$.

On the other hand, from Weyl's theorem (Horn and Johnson, 1986, Theorem 4.3.1), we have

$$\psi_1 \ge \lambda_i'-\lambda_i \ge \psi_N.$$

We have just proved that both $|\psi_1|$ and $|\psi_N|$ are at most $N\varepsilon$, and thus

$$\left|\lambda_i'-\lambda_i\right| \le N\varepsilon.$$

Consequently,

$$\frac{\lambda_i'}{\lambda_i} \le \frac{\lambda_i+N\varepsilon}{\lambda_i}=1+\frac{N\varepsilon}{\lambda_i}.$$

Therefore, we have

$$\ln\frac{\det\Sigma_2}{\det\Sigma_1}=\sum_{i=1}^{N}\ln\frac{\lambda_i'}{\lambda_i} \le \sum_{i=1}^{N}\ln\left(1+\frac{N\varepsilon}{\lambda_i}\right) \le \sum_{i=1}^{N}\frac{N\varepsilon}{\lambda_i} \le \frac{(N)^2\varepsilon}{\lambda_N}$$

where the second to last inequality uses the inequality $\ln(1+x) \le x$ for $x \ge 0$. ■

The following lemmas will be useful to prove Lemma 9.

Lemma 11 ***(Lemma 2.7.1 and Theorem 2.7.2 from Golub and Van Loan 1996)*** *Suppose $Ax = b$ and $(A + \Delta A)y = b + \Delta b$ with $\|\Delta A\| \leq \varepsilon \|A\|$ and $\|\Delta b\| \leq \varepsilon \|b\|$. If $\varepsilon\kappa(A) < 1$, then $A + \Delta A$ is nonsingular, and*

$$\frac{\|y - x\|}{\|x\|} \leq \frac{2\varepsilon\kappa(A)}{1 - \varepsilon\kappa(A)},$$

where $\|\cdot\|$ can be any ℓ_p matrix/vector norm, and $\kappa(A) = \|A\| \left\|A^{-1}\right\|$ is the corresponding condition number.

Lemma 12 ***(A trace inequality of von Neumann 1937)*** *Let A and B be two symmetric matrices of order n, whose singular values are $\xi_1 \geq \xi_2 \geq \cdots \geq \xi_n \geq 0$ and $\zeta_1 \geq \zeta_2 \geq \cdots \geq \zeta_n \geq 0$, respectively. Then*

$$|\mathrm{tr}\,(AB)| \leq \sum_{i=1}^{n} \xi_i \zeta_i.$$

Lemma 13 *Suppose the covariance matrix Σ_1 is non-singular; that is its eigenvalues $\lambda_1 : \lambda_N > 0$. Then*

$$\begin{aligned} \mathrm{tr}\left(\Sigma_1^{-1}\right) &= \sum_{i=1}^{N} \frac{1}{\lambda_i} \leq \frac{N}{\lambda_N} \\ \max_{ij} |\Sigma_1^{-1}(i,j)| &\leq ||\Sigma_1^{-1}||_1 \\ ||\Sigma_1^{-1}||_1 \leq \sqrt{N}||\Sigma_1^{-1}||_2 &= \frac{\sqrt{N}}{\lambda_N}. \end{aligned}$$

Proof We prove the three upper bounds one by one:

1. It is a known fact that the trace of a matrix equals the sum of its eigenvalues. The first equality follows from the observation that the eigenvalues of Σ_1^{-1} are $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_N}$.

2. This inequality follows from the definition of $||\Sigma_1^{-1}||_1$: it is the maximum absolute row sum of the matrix Σ_1^{-1}, and therefore is not less than the largest absolute component of the matrix.

3. It is known that $||A||_1 \leq \sqrt{N}||A||_2$ for any $N \times N$ matrix A (see, eg. theorem 5.6.18 in Horn and Johnson 1986). On the other hand, $||\Sigma_1^{-1}||_2$ equals the largest eigenvalue of Σ_1^{-1}, which is $\frac{1}{\lambda_N}$.

■

Lemma 9 *If $\max_{i,j} |\Sigma_1(i,j) - \Sigma_2(i,j)| \leq \varepsilon$ and $N\varepsilon \left\|\Sigma_1^{-1}\right\|_1 < 1$, then*

$$\mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) - N \leq \frac{2N^3\varepsilon B_\sigma}{\lambda_N^2 - (N)^{1.5}\lambda_N\varepsilon}.$$

Proof The i-th row (or column) of Σ_1^{-1} is the solution to the system of linear equations: $\Sigma_1 x = e_i$ where e_i has $N - 1$ zero components except a 1 in the i-th component. Similarly, the i-th row (or

column) of Σ_2^{-1} is the solution to $\Sigma_2 y = e_i$. Since Σ_1 and Σ_2 differ by at most ε in every component, we have

$$\frac{\|\Sigma_1 - \Sigma_2\|_1}{\|\Sigma_1\|_1} \leq \frac{N\varepsilon}{\|\Sigma_1\|_1}.$$

For convenience, denote the right-hand side above by ε'. It follows from Lemma 11 that

$$\|x - y\|_1 \leq \frac{2\varepsilon'\kappa(\Sigma_1)\,\|x\|_1}{1 - \varepsilon'\kappa(\Sigma_1)}.$$

The above inequality holds for all N possible e values. Note that $\|x-y\|_1$ is the absolute sum of the i-th row (or column) of $\Sigma_1^{-1} - \Sigma_2^{-1}$ for e_i. Let $\psi_1 \geq \psi_2 \geq \cdots \geq \psi_N \geq 0$ be the singular values of $\Sigma_1^{-1} - \Sigma_2^{-1}$. From Geršgorin's theorem, it follows that for all i,

$$\psi_i \leq \max_e \|x-y\|_1 \leq \frac{2\varepsilon'\kappa(\Sigma_1)}{1-\varepsilon'\kappa(\Sigma_1)} \max_e \|x\|_1 = \frac{2\varepsilon'\kappa(\Sigma_1)}{1-\varepsilon'\kappa(\Sigma_1)} \left\|\Sigma_1^{-1}\right\|_1 \tag{12}$$

where $\kappa(\Sigma_1) = \|\Sigma_1\| \left\|\Sigma_1^{-1}\right\|$ the condition number of Σ_1. We can now complete the proof:

$$\begin{aligned}
\operatorname{tr}\left(\Sigma_2^{-1}\Sigma_1\right) - N &= \operatorname{tr}\left((\Sigma_2^{-1} - \Sigma_1^{-1})\Sigma_1\right) && (13)\\
&\leq \sum_{i=1}^{N} \psi_i \lambda_i && (14)\\
&\leq \frac{2\varepsilon'\kappa(\Sigma_1)\left\|\Sigma_1^{-1}\right\|_1}{1-\varepsilon'\kappa(\Sigma_1)} \sum_{i=1}^{N} \lambda_i && (15)\\
&= \frac{2\varepsilon'\kappa(\Sigma_1)\left\|\Sigma_1^{-1}\right\|_1}{1-\varepsilon'\kappa(\Sigma_1)} \operatorname{tr}(\Sigma_1) && (16)\\
&= \frac{2N\varepsilon\left\|\Sigma_1^{-1}\right\|_1^2}{1-N\varepsilon\left\|\Sigma_1^{-1}\right\|_1} \operatorname{tr}(\Sigma_1) && (17)\\
&= \frac{2N^2 B_\sigma \varepsilon\left\|\Sigma_1^{-1}\right\|_1^2}{1-N\varepsilon\left\|\Sigma_1^{-1}\right\|_1}, && (18)\\
&\leq \frac{2N^3\varepsilon B_\sigma}{\lambda_N^2 - (N)^{1.5}\lambda_N\varepsilon}. && (19)
\end{aligned}$$

The first equality (Equation 13) is due to the identity $\operatorname{tr}\left(\Sigma_1^{-1}\Sigma_1\right) = \operatorname{tr}(I) = N$, and the first inequality (Equation 14) is a direct application of von Neumann's inequality (Lemma 12) which can be used since the eigenvalues λ_i are also the singular values in this case. The second inequality (Equation 15) follows from the result of Equation 12, the second equality (Equation 16) follows by the definition of matrix traces, and the third equality (Equation 17) is obtained by noting that $\kappa(\Sigma_1) = \|\Sigma_1\|_1 \left\|\Sigma_1^{-1}\right\|_1$. Since each term in the covariance matrix is known to be bounded by B_σ then the trace is bounded by NB_σ which allows us to generate the fourth equality (Equation 18). The final result is obtained using the result of Lemma 13.

■

Lemma 14 ***(Theorem from Kullback 1967)*** *Let p_1 and p_2 be two probability density functions defined over $\mathcal{X}$. Define*

$$\Omega = \{x \in \mathcal{X} \mid p_1(x) \geq p_2(x)\}.$$

If p_1 and p_2 are both measurable (integrable) over Ω, then

$$d_{KL}(p_1 \,\|\, p_2) \geq \frac{1}{8} \|p_1 - p_2\|_1^2.$$

References

Pieter Abbeel and Andrew Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 1–8, 2005.

Justin Boyan and Andrew Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems (NIPS) 7*, pages 369–376, 1995.

Ronen I. Brafman and Moshe Tennenholtz. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

Emma Brunskill, Bethany R. Leffler, Lihong Li, Michael L. Littman, and Nicholas Roy. CORL: A continuous-state offset-dynamics reinforcement learner. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 53–61, 2008.

Jeffrey B. Burl. *Linear Optimal Control*. Prentice Hall, 1998. ISBN 9780201808681.

Pablo Castro and Doina Precup. Using linear programming for Bayesian exploration in Markov decision processes. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2437–2442, 2007.

Chee-Seng Chow and John N. Tsitsiklis. The complexity of dynamic programming. *Journal of Complexity*, 5(4):466–488, 1989.

Chee-Seng Chow and John N. Tsitsiklis. An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control*, 36(8):898–914, 1991.

Finale Doshi, Joelle Pineau, and Nicholas Roy. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 256–263, 2008.

Alain Dutech, Timothy Edmunds, Jelle Kok, Michail Lagoudakis, Michael L. Littman, Martin Riedmiller, Bryan Russell, Bruno Scherrer, Richard Sutton, Stephan Timmer, Nikos Vlassis, Adam White, and Shimon Whiteson. Reinforcement learning benchmarks and bake-offs II. In *Advances in Neural Information Processing Systems (NIPS) 17 Workshop*, 2005.

Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: A WiFi-Based Vehicular Content Delivery Network. In *Proceedings of the 14th Conference on Mobile Computing and Networking (MOBICOM)*, pages 199–210, 2008.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996. ISBN 0-801-85414-8.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1986. ISBN 0-521-38632-2.

Eric Horvitz, Johnson Apacible, Raman Sarin, and Lin Liao. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 275–283, 2005.

Nicholas K. Jong and P. Stone. Model-based exploration in continuous state spaces. In *Proceedings of the 7th Symposium on Abstraction, Reformulation, and Approximation (SARA)*, pages 258–272, 2007.

Sham Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, University College London, 2003.

Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2–3):209–232, 2002.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML)*, pages 282–293, 2006.

Solomon Kullback. A lower bound for discrimination in terms of variation. *IEEE Transactions on Information Theory*, 13(1):126–127, January 1967.

Branislav Kveton and Milos Hauskrecht. Solving factored MDPs with exponential-family transition models. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 114–120, 2006.

M.G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

Bethany R. Leffler, Michael L. Littman, and Timothy Edmunds. Efficient reinforcement learning with relocatable action models. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI)*, pages 572–577, 2007.

Lihong Li. *A Unifying Framework for Computational Reinforcement Learning Theory*. PhD thesis, Rutgers University, New Brunswick, NJ, 2009.

Lihong Li, Michael L. Littman, and Thomas J. Walsh. Knows what it knows: A framework for self-aware learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 568–575, 2008.

Janusz Marecki and Milind Tambe. Towards faster planning with continuous resources in stochastic domains. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*, pages 1049–1055, 2008.

Andrew Ng, H.Jin Kim, Michael Jordan, and Shankar Sastry. Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS) 16*, pages 799–806, 2004.

Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 697–704, 2006.

Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayesian reinforcement learning in continuous POMDPs with application to robot navigation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 2845–2851, 2008.

Alexander L. Strehl and Michael L. Littman. Online linear regression and its application to model-based reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS) 20*, pages 1417–1424, 2008.

Alexander L. Strehl, Lihong Li, and Michael L. Littman. Incremental model-based learners with formal learning-time guarantees. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 485–492, 2006.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

Gerald J. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

John von Neumann. Some matrix-inequalities and metrization of matrix-space. *Tomsk University Review*, 1:286–300, 1937.

Christopher J.C.H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, University of Cambridge, United Kingdom, 1989.

Journal of Machine Learning Research 10 (2009) 1989-2012 Submitted 7/08; Revised 11/08; Published 9/09

Fast Approximate kNN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection

Jie Chen JCHEN@CS.UMN.EDU
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455, USA

Haw-ren Fang HRFANG@MCS.ANL.GOV
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439, USA

Yousef Saad SAAD@CS.UMN.EDU
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455, USA

Editor: Sanjoy Dasgupta

Abstract

Nearest neighbor graphs are widely used in data mining and machine learning. A brute-force method to compute the exact kNN graph takes $\Theta(dn^2)$ time for n data points in the d dimensional Euclidean space. We propose two divide and conquer methods for computing an approximate kNN graph in $\Theta(dn^t)$ time for high dimensional data (large d). The exponent $t \in (1,2)$ is an increasing function of an internal parameter α which governs the size of the common region in the divide step. Experiments show that a high quality graph can usually be obtained with small overlaps, that is, for small values of t. A few of the practical details of the algorithms are as follows. First, the divide step uses an inexpensive Lanczos procedure to perform recursive spectral bisection. After each conquer step, an additional refinement step is performed to improve the accuracy of the graph. Finally, a hash table is used to avoid repeating distance calculations during the divide and conquer process. The combination of these techniques is shown to yield quite effective algorithms for building kNN graphs.

Keywords: nearest neighbors graph, high dimensional data, divide and conquer, Lanczos algorithm, spectral method

1. Introduction

Building nearest neighbor graphs is often a necessary step when dealing with problems arising from applications in such areas as data mining (Brito et al., 1997; Dasarathy, 2002), manifold learning (Belkin and Niyogi, 2003; Roweis and Saul, 2000; Saul and Roweis, 2003; Tenenbaum et al., 2000), robot motion planning (Choset et al., 2005), and computer graphics (Sankaranarayanan et al., 2007). Given a set of n data points $X = \{x_1, \ldots, x_n\}$, a nearest neighbor graph consists of the vertex set X and an edge set which is a subset of $X \times X$. The edges are defined based on a *proximity* measure $\rho(x_i, x_j)$ between two data points x_i and x_j, where a small ρ value means that the two points are

close. Two types of nearest neighbor graphs (Belkin and Niyogi, 2003; He and Niyogi, 2004) are often used:

1. ε-graph: This is an undirected graph whose edge set consists of pairs (x_i, x_j) such that $\rho(x_i, x_j)$ is less than some pre-defined threshold $\varepsilon \in \mathbb{R}^+$.

2. kNN graph: This is a directed graph (in general). There is an edge from x_i to x_j if and only if $\rho(x_i, x_j)$ is among the k smallest elements of the set $\{\rho(x_i, x_\ell) \mid \ell = 1, \ldots, i-1, i+1, \ldots, n\}$.

The ε-graph is geometrically motivated, and many efficient algorithms have been proposed for computing it (see, e.g., Bentley et al., 1977; Chazelle, 1983). However, the ε-graph easily results in disconnected components (Belkin and Niyogi, 2003) and it is difficult to find a good value of ε which yields graphs with an appropriate number of edges. Hence, they are not suitable in many situations. On the other hand, kNN graphs have been shown to be especially useful in practice. Therefore, this paper will focus on the construction of kNN graphs, for the case when $\rho(\cdot, \cdot)$ is the Euclidean distance between two points in the $\mathbb{R}^d$.

When $k = 1$, only the nearest neighbor for each data point is considered. This particular case, the *all nearest neighbors* problem, has been extensively studied in the literature. To compute the 1NN graph, Bentley (1980) proposed a multidimensional divide-and-conquer method that takes $O(n \log^{d-1} n)$ time, Clarkson (1983) presented a randomized algorithm with expected $O(c^d n \log n)$ time (for some constant c), and Vaidya (1989) introduced a deterministic worst-case $O((c'd)^d n \log n)$ time algorithm (for some constant c'). These algorithms are generally adaptable to $k > 1$. Thus, Paredes et al. (2006) presented a method to build a kNN graph, which was empirically studied and reported to require $O(n^{1.27})$ distance calculations for low dimensional data and $O(n^{1.90})$ calculations for high dimensional data. Meanwhile, several parallel algorithms have also been developed (Callahan, 1993; Callahan and Rao Kosaraju, 1995; Plaku and Kavraki, 2007). Despite a rich existing literature, efficient algorithms for high dimensional data are still under-explored. In this paper we propose two methods that are especially effective in dealing with high dimensional data.

Note that the problem of constructing a kNN graph is different from the problem of *nearest neighbor(s) search* (see, e.g., Indyk, 2004; Shakhnarovich et al., 2006, and references therein), where given a set of data points, the task is to find the k nearest points for any query point. Usually, the nearest neighbors search problem is handled by first building a data structure (such as a search tree) for the given points in a preprocessing phase. Then, queries can be answered efficiently by exploiting the search structure. Of course, the construction of a kNN graph can be viewed as a nearest neighbors search problem where each data point itself is a query. However, existing search methods in general suffer from an unfavorable trade-off between the complexity of the search structure and the accuracy of the query retrieval: either the construction of the search structure is expensive, or accurate searches are costly. The spill-tree (sp-tree) data structure (Liu et al., 2004) is shown to be empirically effective in answering queries, but since some heuristics (such as the hybrid scheme) are introduced to ensure search accuracy, the construction time cost is difficult to theoretically analyze. On the other hand, many $(1+\varepsilon)$ nearest neighbor search methods[1] have been proposed with guaranteed complexity bounds (Indyk, 2004). These methods report a point within $(1+\varepsilon)$ times the distance from the query to the actual nearest neighbor. Kleinberg (1997) presented two algorithms. The first one has a $O((d \log^2 d)(d + \log n))$ query time complexity but requires a data structure of

1. The ε parameter here is different from that in the "ε-graph".

size $O(n\log d)^{2d}$, whereas the second algorithm uses a nearly linear (to dn) data structure and responds to a query in $O(n+d\log^3 n)$ time. By using locality sensitive hashing, Indyk and Motwani (1998) gave an algorithm that uses $O(n^{1+1/(1+\varepsilon)}+dn)$ preprocessing time and requires $O(dn^{1/(1+\varepsilon)})$ query time. Another algorithm given in Indyk and Motwani (1998) is a $O(d\,\mathrm{poly}\log(dn))$ search algorithm that uses a $O(n(1/\varepsilon)^{O(d)}\,\mathrm{poly}\log(dn))$ data structure. Kushilevitza et al. (2000) proposed a $O((dn)^2(n\log(d\log n/\varepsilon))^{O(\varepsilon^{-2})}\,\mathrm{poly}(1/\varepsilon)\,\mathrm{poly}\log(dn/\varepsilon))$ data structure that can answer a query in $O(d^2\,\mathrm{poly}(1/\varepsilon)\,\mathrm{poly}\log(dn/\varepsilon))$ time. In general, most of the given bounds are theoretical, and have an inverse dependence on ε, indicating expensive costs when ε is small. Some of these methods can potentially be applied for the purpose of efficient kNN graph construction, but many aspects, such as appropriate choices of the parameters that control ε, need to be carefully considered before a practically effective algorithm is derived. Considerations of this nature will not be explored in the present paper.

The rest of the paper is organized as follows. Section 2 proposes two methods for computing approximate kNN graphs, and Section 3 analyzes their time complexities. Then, we show a few experiments to demonstrate the effectiveness of the methods in Section 4, and discuss two applications in Section 5. Finally, conclusions are given in Section 6.

2. Divide and Conquer kNN

Our general framework for computing an approximate kNN graph is as follows: We divide the set of data points into subsets (possibly with overlaps), recursively compute the (approximate) kNN graphs for the subsets, then conquer the results into a final kNN graph. This divide and conquer framework can clearly be separated in two distinct sub-problems: how to divide and how to conquer. The conquer step is simple: If a data point belongs to more than one subsets, then its k nearest neighbors are selected from its neighbors in each subset. However, the divide step can be implemented in many different ways, resulting in different qualities of graphs. In what follows two methods are proposed which are based on a spectral bisection (Boley, 1998; Juhász and Mályusz, 1980; Tritchler et al., 2005) of the graph obtained from an inexpensive Lanczos procedure (Lanczos, 1950).

2.1 Spectral Bisection

Consider the data matrix

$$X = [x_1, \ldots, x_n] \in \mathbb{R}^{d\times n}$$

where each column x_i represents a data point in $\mathbb{R}^d$. When the context is clear, we will use the same symbol X to also denote the set of data points. The data matrix X is centered to yield the matrix:

$$\hat{X} = [\hat{x}_1, \ldots, \hat{x}_n] = X - ce^T,$$

where c is the centroid and e is a column of all ones. A typical spectral bisection technique splits $\hat{X}$ into halves using a hyperplane. Let (σ, u, v) denote the largest singular triplet of $\hat{X}$ with

$$u^T\hat{X} = \sigma v^T. \tag{1}$$

Then, the hyperplane is defined as $\langle u, x-c\rangle = 0$, that is, it splits the set of data points into two subsets:

$$X_+ = \{x_i \mid u^T\hat{x}_i \geq 0\} \quad \text{and} \quad X_- = \{x_i \mid u^T\hat{x}_i < 0\}.$$

This hyperplane maximizes the sum of squared distances between the centered points $\hat{x}_i$ to the hyperplane that passes through the centroid. This is because for any hyperplane $\langle w, x-c\rangle = 0$, where w is a unit vector, the squared sum is

$$\sum_{i=1}^{n} (w^T \hat{x}_i)^2 = \left\| w^T \hat{X} \right\|_2^2 \leq \left\| \hat{X} \right\|_2^2 = \sigma^2,$$

while $w = u$ achieves the equality.

By a standard property of the SVD (Equation 1), this bisection technique is equivalent to splitting the set by the following criterion:

$$X_+ = \{x_i \mid v_i \geq 0\} \quad \text{and} \quad X_- = \{x_i \mid v_i < 0\}, \tag{2}$$

where v_i is the i-th entry of the right singular vector v. If it is preferred that the sizes of the two subsets be balanced, an alternative is to replace the above criterion by

$$X_+ = \{x_i \mid v_i \geq \mathrm{m}(v)\} \quad \text{and} \quad X_- = \{x_i \mid v_i < \mathrm{m}(v)\}, \tag{3}$$

where $\mathrm{m}(v)$ represents the median of the entries of v.

The largest singular triplet (σ, u, v) of $\hat{X}$ can be computed using the Lanczos algorithm (Lanczos, 1950; Berry, 1992). In short, we first compute an orthonormal basis of the Krylov subspace $\mathrm{span}\{q_1, (\hat{X}^T\hat{X})q_1, \ldots, (\hat{X}^T\hat{X})^{s-1}q_1\}$ for an arbitrary initial unit vector q_1 and a small integer s. Let the basis vectors form an orthogonal matrix

$$Q_s = [q_1, \ldots, q_s].$$

An equality resulting from this computation is

$$Q_s^T (\hat{X}^T\hat{X}) Q_s = T_s,$$

where T_s is a symmetric tridiagonal matrix of size $s \times s$. Then we compute the largest eigenvalue $\theta^{(s)}$ and corresponding eigenvector y_s of T_s:

$$T_s y_s = \theta^{(s)} y_s.$$

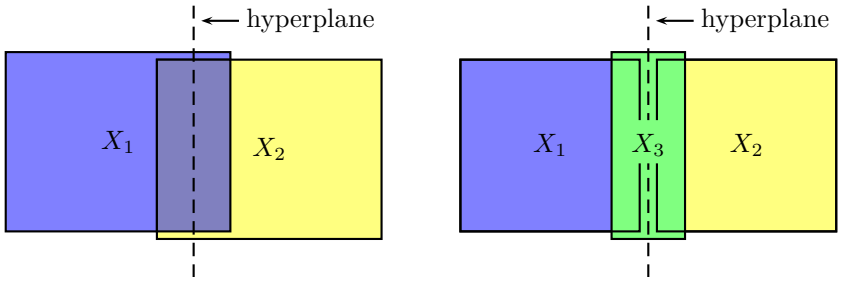
Therefore, $\theta^{(s)}$ is an approximation to the square of the largest singular value σ, while the vector $\tilde{v}_s \equiv Q_s y_s$ is an approximation of the right singular vector v of $\hat{X}$. The acute angle $\angle(\tilde{v}_s, v)$, between $\tilde{v}_s$ and v decays rapidly with s, as is shown in an error bound established in Saad (1980):

$$\sin\angle(\tilde{v}_s, v) \leq \frac{K}{C_{s-1}(1+\gamma_1)}$$

where K is a constant, C_k denotes the Chebyshev polynomial of degree k of the first kind, and $\gamma_1 = (\lambda_1 - \lambda_2)/(\lambda_2 - \lambda_n)$ in which the λ_i's denote the eigenvalues of $\hat{X}^T\hat{X}$, that is, the squared singular values of $\hat{X}$, ordered decreasingly. Note that we have $C_{s-1}(1+\gamma_1) = \cosh[(s-1)\cosh^{-1}(1+\gamma_1)]$, in which cosh is the hyperbolic cosine. Therefore, a small value of s will generally suffice to yield an accurate approximation. Note that an exact singular vector is not needed to perform a bisection, so we usually set a fixed value, say $s = 5$, for this purpose. The computation of the orthonormal basis takes time $\Theta(sdn)$, while the time to compute $\theta^{(s)}$ and y_s is negligible, since T_s is symmetric tridiagonal and s is very small. Hence the time for computing the largest singular triplet of $\hat{X}$ is bounded by $O(sdn)$.

2.2 The Divide Step: Two Methods

Based on the above general bisection technique, we propose two ways to perform the divide step for computing an approximate kNN graph. The first, called the *overlap* method, divides the current set into two overlapping subsets. The second, called the *glue* method, divides the current set into two disjoint subsets, and uses a third set called the gluing set to help merge the two resulting disjoint kNN graphs in the conquer phase. See Figure 1. Both methods create a common region, whose size is an α-portion of that of the current set, surrounding the dividing hyperplane. Details are given next.



(a) The overlap method. (b) The glue method.

Figure 1: Two methods to divide a set X into subsets. The size of the common region in the middle surrounding the hyperplane is an α-portion of the size of X.

2.2.1 THE OVERLAP METHOD

In this method, we divide the set X into two overlapping subsets X_1 and X_2:

$$\begin{cases} X_1 \cup X_2 = X, \\ X_1 \cap X_2 \neq \emptyset. \end{cases}$$

Since σv_i is the signed distance from $\hat{x}_i$ to the hyperplane, let the set V be defined as

$$V = \{|v_i| \mid i = 1, 2, \ldots, n\}.$$

Then we use the following criterion to form X_1 and X_2:

$$X_1 = \{x_i \mid v_i \geq -h_\alpha(V)\} \quad \text{and} \quad X_2 = \{x_i \mid v_i < h_\alpha(V)\}, \tag{4}$$

where $h_\alpha(\cdot)$ is a function which returns the element that is only larger than $(100\alpha)\%$ of the elements in the input set. The purpose of criterion (4) is to ensure that the overlap of the two subsets consists of $(100\alpha)\%$ of all the data, that is, that[2]

$$|X_1 \cap X_2| = \lceil \alpha |X| \rceil .$$

2. Here, we assume that the distances between points and the hyperplane are all different. Ties are broken arbitrarily. The same is done for the glue method.

2.2.2 THE GLUE METHOD

In this method, we divide the set X into two disjoint subsets X_1 and X_2 with a gluing subset X_3:

$$\begin{cases} X_1 \cup X_2 = X, \\ X_1 \cap X_2 = \emptyset, \\ X_1 \cap X_3 \neq \emptyset, \\ X_2 \cap X_3 \neq \emptyset. \end{cases}$$

The criterion to build these subsets is as follows:

$$X_1 = \{x_i \mid v_i \geq 0\}, \quad X_2 = \{x_i \mid v_i < 0\},$$
$$X_3 = \{x_i \mid -h_\alpha(V) \leq v_i < h_\alpha(V)\}.$$

Note that the gluing subset X_3 in this method is exactly the intersection of the two subsets in the overlap method. Hence, X_3 also contains $(100\alpha)\%$ of the data.

2.3 Refinement

In order to improve the quality of the resulting graph, during each recursion after the conquer step, the graph can be refined at a small cost. The idea is to update the k nearest neighbors for each point by selecting from a pool consisting of its neighbors and the neighbors of its neighbors. Formally, if $N(x)$ is the set of current nearest neighbors of x before refinement, then, for each point x, we re-select its k nearest neighbors from

$$N(x) \cup \left(\bigcup_{z \in N(x)} N(z) \right).$$

2.4 The Algorithms

We are ready to present the complete algorithms for both methods; see Algorithms 1 and 2. These algorithms share many similarities: They both fall in the framework of divide and conquer; they both call the brute-force procedure kNN-BRUTEFORCE to compute the graph when the size of the set is smaller than a threshold (n_k); they both recursively call themselves on smaller subsets; and they both employ a CONQUER procedure to merge the graphs computed for the subsets and a REFINE procedure to refine the graph during each recursion. The difference is that Algorithm 1 calls DIVIDE-OVERLAP to divide the set into two subsets (Section 2.2.1), while Algorithm 2 calls DIVIDE-GLUE to divide the set into three subsets (Section 2.2.2). For the sake of completeness, pseudocodes of all the mentioned procedures are given in Appendix A.

2.5 Storing Computed Distances in a Hash Table

The brute-force method computes $\Theta(n^2)$ pairs of distances, each of which takes $\Theta(d)$ time. One advantage of the methods presented in this paper over brute-force methods is that the distance calculations can be significantly reduced thanks to the divide and conquer approach. The distances are needed/computed in: (1) the kNN-BRUTEFORCE procedure which computes all the pairwise distances and selects the k smallest ones for each point, (2) the CONQUER procedure which selects

Algorithm 1 Approximate kNN Graph Construction: The Overlap Method

function $G = k$NN-OVERLAP(X, k, α)
  if $|X| < n_k$ **then**
    $G \leftarrow k$NN-BRUTEFORCE(X, k)
  else
    $(X_1, X_2) \leftarrow$ DIVIDE-OVERLAP(X, α) ▷ Section 2.2.1
    $G_1 \leftarrow k$NN-OVERLAP(X_1, k, α)
    $G_2 \leftarrow k$NN-OVERLAP(X_2, k, α)
    $G \leftarrow$ CONQUER(G_1, G_2, k) ▷ Section 2, beginning
    REFINE(G, k) ▷ Section 2.3
  end if
end function

Algorithm 2 Approximate kNN Graph Construction: The Glue Method

function $G = k$NN-GLUE(X, k, α)
  if $|X| < n_k$ **then**
    $G \leftarrow k$NN-BRUTEFORCE(X, k)
  else
    $(X_1, X_2, X_3) \leftarrow$ DIVIDE-GLUE(X, α) ▷ Section 2.2.2
    $G_1 \leftarrow k$NN-GLUE(X_1, k, α)
    $G_2 \leftarrow k$NN-GLUE(X_2, k, α)
    $G_3 \leftarrow k$NN-GLUE(X_3, k, α)
    $G \leftarrow$ CONQUER(G_1, G_2, G_3, k) ▷ Section 2, beginning
    REFINE(G, k) ▷ Section 2.3
  end if
end function

k smallest distances from at most $2k$ candidates for each point, and (3) the REFINE procedure which selects the k smallest distances from at most $k + k^2$ candidates for each point. Many of the distances computed from kNN-BRUTEFORCE and REFINE are reused in CONQUER and REFINE, with probably more than once for some pairs. A naive way is to allocate memory for an $n \times n$ matrix that stores all the computed distances to avoid duplicate calculations. However this consumes too much memory and is not necessary. A better approach is to use a hash table to store the computed distances. This will save a significant amount of memory, as a later experiment shows that only a small portion of the n^2 pairs are actually computed. Furthermore the computational time will not be affected since hash tables are efficient for both retrieving wanted items from and inserting new items into the table (we do not need the delete operation).

The ideal hash function maps (the distance between) a pair of data points (x_i, x_j) to a bucket such that the probability of collision is low. For simplicity of implementations, we use a hash function that maps the key (x_i, x_j) to i (and at the same time to j). Collisions easily occur since many distances between the point x_i and other points are computed during the whole process. However, this naive hashing has already shown rather appealing results in run time. More elaborate implementations should consider more effective hashing (e.g., double hashing) schemes.

3. Complexity Analysis

A thorough analysis shows that the time complexities for the overlap method and the glue method are sub-quadratic (in n), and the glue method is always asymptotically faster than the overlap method. To this end we assume that in each divide step the subsets X_1 and X_2 (in both methods) are balanced. This assumption can always be satisfied by using the Equation (3) to bisect the data set, instead of the criterion (2). Hence, the time complexity $T_{\rm o}$ for the overlap method and $T_{\rm g}$ for the glue method satisfy the following recurrence relations:

$$T_{\rm o}(n) = 2T_{\rm o}((1+\alpha)n/2) + f(n), \tag{5}$$

$$T_{\rm g}(n) = 2T_{\rm g}(n/2) + T_{\rm g}(\alpha n) + f(n), \tag{6}$$

where $f(n)$ is the combined time for the divide, conquer, and refine steps.

3.1 The Complexity of f

The function $f(n)$ consists of the following three components.

3.1.1 THE TIME FOR THE DIVIDE STEP

This includes the time to compute the largest right singular vector v of the centered matrix $\hat{X}$, and the time to divide points into subsets X_1 and X_2 (in the overlap method) or subsets X_1, X_2 and X_3 (in the glue method). The former has been shown to be $O(sdn)$ in Section 2.1, where the number of Lanczos steps $s = 5$ is fixed in the implementation, while for the latter we can use a linear time selection method to find the value $h_\alpha(V)$. Therefore the overall time for this step is $O(dn)$.

3.1.2 THE TIME FOR THE CONQUER STEP

This step only involves the points in $X_1 \cap X_2$ in the overlap method or X_3 in the glue method. For each of the αn points in these subsets, k nearest neighbors are chosen from at most $2k$ candidates. Therefore the time is $O(k\alpha n)$.

3.1.3 THE TIME FOR THE REFINE STEP

For each point, k nearest neighbors are chosen from at most $k+k^2$ candidates. If all these distances need to be computed, the overall time is $O(k^2dn)$. To the other extreme, if none of them are computed, the factor d can be omitted, which results in $O(k^2n)$. In practice, by using a hash table, only a very small fraction of the $k+k^2$ distances are actually computed in this step; see also Table 3. Hence, the best cost estimate for this step is $O(k^2n)$.

Indeed, the ultimate useful information from this estimate is that the time for the refinement is much less than that for the division (which has a $O(dn)$ cost), or at best is of the same order as the latter. This can be seen from another perspective: Let the number of neighbors k be a constant. Then even if all the distances are computed, the time is $O(k^2dn) = O(dn)$.

From the above three components and the fact that the dimension d dominates k, we conclude that $f(n)$ is bounded by $O(dn)$.

3.2 The Complexities of T_o and T_g

By substituting $f(n) = O(dn)$ into (5) and (6), we derive the closed form solutions to $T_o(n)$ and $T_g(n)$, which are stated in the following two theorems.

Theorem 1 *The time complexity for the overlap method is*

$$T_o(n) = \Theta(dn^{t_o}),$$

where

$$t_o = \log_{2/(1+\alpha)} 2 = \frac{1}{1-\log_2(1+\alpha)}.$$

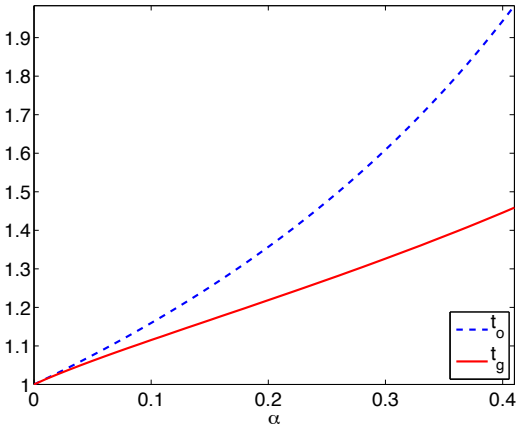
Theorem 2 *The time complexity for the glue method is*

$$T_g(n) = \Theta(dn^{t_g}/\alpha),$$

where t_g is the solution to the equation

$$\frac{2}{2^t} + \alpha^t = 1.$$

The proofs will be given in Appendix B. Figure 2 plots the two curves of t_o and t_g as functions of α, together with a table that lists some of their values. Figure 2 suggests that the glue method is asymptotically faster than the overlap method. Indeed, this is true for any choice of α.



α	0.05	0.10	0.15	0.20	0.25	0.30
t_o	1.08	1.16	1.25	1.36	1.47	1.61
t_g	1.06	1.12	1.17	1.22	1.27	1.33

Figure 2: The exponents t_o and t_g as functions of α.

Theorem 3 *When $0 < \alpha < 1$, the exponents t_o in Theorem 1 and t_g in Theorem 2 obey the following relation:*

$$1 < t_g < t_o. \tag{7}$$

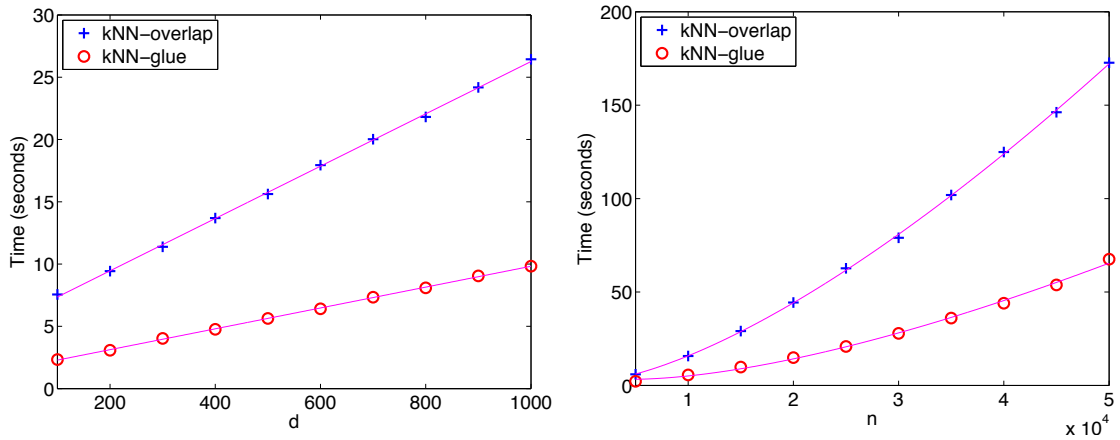
The proof of the above theorem will be given in Appendix B. We remark that when $\alpha > \sqrt{2} - 1 \approx 0.41$, $t_o > 2$. In this situation the overlap method becomes asymptotically slower than the brute-force method. Similarly, when $\alpha > 1/\sqrt{2} \approx 0.71$, $t_g > 2$. Hence, a large α (> 0.41) is not useful in practice.

4. Experiments

In this section we show a few experimental results to illustrate the running times of the two proposed methods compared with the brute-force method, and the qualities of the resulting graphs. The experiments were performed under a Linux workstation with two P4 3.20GHz CPUs and 2GB memory. The algorithms were all implemented using C/C++, and the programs were compiled using g++ with -O2 level optimization. The divide and conquer methods were implemented according to Algorithms 1 and 2, and the brute-force method was exactly as in the procedure kNN-BRUTEFORCE given in Appendix A.

4.1 Running Time

Figure 3 plots the running times versus the dimension d and the number of data points n on a synthetic data set. Since the distribution of the data should have little impact on the running times of the methods, we used random data (drawn from the uniform distribution over $[0,1]^d$) for this experiment. From Figure 3(a), it is clear that the running time is linear with respect to the dimension d. This is expected from the complexity analysis. In Figure 3(b), we used $\alpha = 0.2$, which corresponds to the theoretical values $t_o = 1.36$ and $t_g = 1.22$. We used curves in the form $c_1 n^{1.36} + c_2 n + c_3$ and $c_4 n^{1.22} + c_5 n + c_6$ to fit the running times of the overlap method and the glue method, respectively. The fitted curves are also shown in Figure 3. It can be seen that the experimental results match the theoretical analysis quite well.



(a) $n = 10000, \alpha = 0.2, d$ varying (b) $d = 500, \alpha = 0.2, n$ varying

Figure 3: The running times for randomly generated data.

4.2 Quality

In another experiment, we used four real-life data sets to test the qualities of the resulting kNN graphs: The FREY[3] face video frames (Roweis and Saul, 2000), the extYaleB[4] database (Lee et al.,

3. FREY can be found at http://www.cs.toronto.edu/~roweis/data.html.
4. extYaleB can be found at http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html.

2005), the MNIST[5] digit images (LeCun et al., 1998), and the PIE[6] face database (Sim et al., 2003). These data sets are all image sets that are widely used in the literature in the areas of face recognition, dimensionality reduction, etc. For MNIST, we used only the test set. Table 1 gives some characteristics of the data. The number of images is equal to n and the size of each image, that is, the total number of pixels for each image, is the dimension d. The dimensions vary from about 500 to 10,000. Since some of these data sets were also used later to illustrate the practical usefulness of our methods in real applications, for each one we used a specific k value that was used in the experiments of past publications. These values are typically around 10.

	FREY	extYaleB	MNIST	PIE
# imgs (n)	1,965	2,414	10,000	11,554
img size (d)	20×28	84×96	28×28	32×32

Table 1: Image data sets.

The qualities of the resulting graphs versus the running times are plotted in Figure 4. Each plotted point in the figure corresponds to a choice of α ($= 0.05, 0.10, 0.15, 0.20, 0.25, 0.30$). The running times of the brute-force method are also indicated. We use two quality criteria: *accuracy* and *average rank*. The *accuracy* of an approximate kNN graph G' (with regard to the exact graph G) is defined as

$$\text{accuracy}(G') = \frac{|E(G') \cap E(G)|}{|E(G)|},$$

where $E(\cdot)$ means the set of directed edges in the graph. Thus, the accuracy is within the range $[0,1]$, and higher accuracy means better quality. The rank $r_u(v)$ of a vertex v with regard to a vertex u is the position of v in the vertex list sorted in ascending order of the distance to u. (By default, the rank of the nearest neighbor of u is 1.) Thus, the *average rank* is defined as

$$\text{ave-rank}(G') = \frac{1}{kn} \sum_u \sum_{v \in N(u)} r_u(v),$$

where $N(u)$ means the neighborhood of u in the graph G'. The exact kNN graph has the average rank $(1+k)/2$.

It can be seen from Figure 4 that the qualities of the resulting graphs exhibit similar trends by using both measures. Take the graph accuracy for example. The larger α, the more accurate the resulting graph. However, larger values of α lead to more time-consuming runs. In addition, the glue method is much faster than the overlap method for the same α, while the latter yields more accurate graphs than the former. The two methods are both significantly faster than the brute-force method when an appropriate α is chosen, and they can yield high quality graphs even when α is small.

It is interesting to note that in all the plots, the red circle-curve and the blue plus-curve roughly overlap. This similar quality-time trade-off seems to suggest that neither of the method is superior to the other one; the only difference is the α value used to achieve the desired quality. However, we note that different approximate graphs can yield the same accuracy value or average rank value, hence the actual quality of the graph depends on real applications.

5. MNIST can be found at `http://yann.lecun.com/exdb/mnist`.
6. PIE can be found at `http://www.cs.uiuc.edu/homes/dengcai2/Data/FaceData.html`.

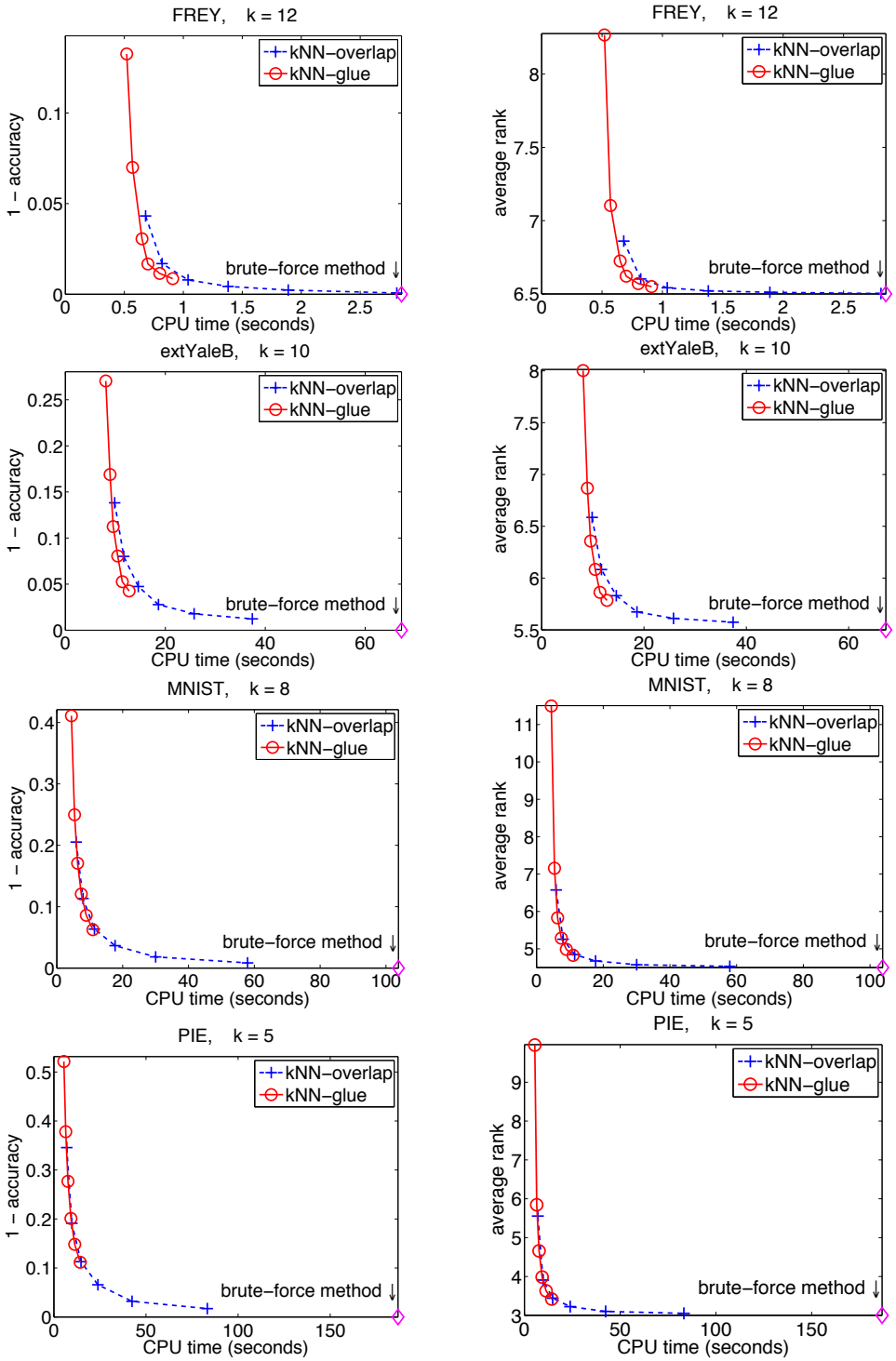


Figure 4: Graph quality versus running time for different data sets. Each row of the plots corresponds to one data set. The left column of plots shows the 1−accuracy measure. The right column shows the average-rank measure. Each plotted point corresponds to a choice of α. From left to right on each curve, the α values are $0.05, 0.10, 0.15, 0.20, 0.25, 0.30$, respectively.

4.3 Distance Calculations and Refinements

We also report the percentage of the distance calculations, which is one of the dominant costs of the graph construction. The superior efficiency of our methods lies in two facts. First, for n data points, the brute-force method must compute distances between $n(n-1)/2$ pairs of points, while our methods compute only a small fraction of this number. Second, the use of a (hash) table brings the benefit of avoiding repeating distance calculations if the evaluations of the same distance are required multiple times. Tables 2 and 3 confirm these two claims. As expected, the larger the common region (α) is, the more distances need to be calculated, while the benefit of the hashing becomes more significant. It can also be seen that the savings in distance calculations are more remarkable when n becomes larger and k becomes smaller, for example, $k = 5$ for the data set PIE.

	FREY ($k = 12$)		extYaleB ($k = 10$)		MNIST ($k = 8$)		PIE ($k = 5$)	
α	overlap	glue	overlap	glue	overlap	glue	overlap	glue
0.05	6.07%	5.10%	5.13%	4.42%	1.19%	0.94%	0.45%	0.35%
0.10	6.55%	5.80%	5.58%	5.08%	1.42%	1.22%	0.60%	0.47%
0.15	7.48%	6.35%	6.05%	5.46%	1.74%	1.36%	0.78%	0.56%
0.20	8.28%	6.57%	6.66%	5.66%	2.20%	1.52%	1.06%	0.66%
0.25	9.69%	7.00%	7.36%	6.02%	2.92%	1.71%	1.48%	0.77%
0.30	11.48%	7.46%	8.34%	6.26%	4.04%	1.91%	2.07%	0.90%

Table 2: Percentages of distance calculations (with respect to $n(n-1)/2$), for different data sets, different methods, and different α's.

	FREY ($k = 12$)		extYaleB ($k = 10$)		MNIST ($k = 8$)		PIE ($k = 5$)	
α	overlap	glue	overlap	glue	overlap	glue	overlap	glue
0.05	9.44%	12.15%	11.46%	14.80%	9.56%	11.55%	6.85%	7.01%
0.10	7.07%	12.09%	8.46%	14.36%	6.82%	11.76%	5.27%	8.14%
0.05	5.18%	10.81%	6.25%	12.79%	4.66%	10.17%	3.52%	7.55%
0.20	3.65%	9.12%	4.35%	11.04%	2.98%	8.49%	2.15%	6.40%
0.05	2.62%	7.52%	2.90%	9.55%	1.77%	6.85%	1.23%	5.19%
0.30	1.83%	6.44%	1.88%	7.89%	0.96%	5.41%	0.62%	4.13%

Table 3: Percentages of actual distance calculations with respect to the total number of needed distances in the refine step, for different data sets, different methods, and different α's. This is equivalent to the failure rate of hash table lookups.

The final experiment illustrates the importance of the refine step. Figure 5 shows the decrease in quality if the REFINE procedure is not invoked. The refinement greatly improves the accuracy of the approximate graph (especially for α less than 0.2) at some additional cost in the execution time. This additional expense is worthwhile if the goal is to compute a high quality kNN graph.

5. Applications

kNN graphs have been widely used in various data mining and machine learning applications. This section discusses two scenarios where the approximate kNN graphs resulting from our proposed techniques can provide an effective replacement for the exact kNN graph.

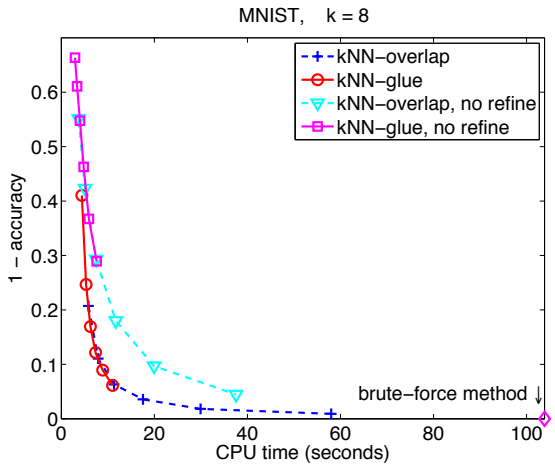


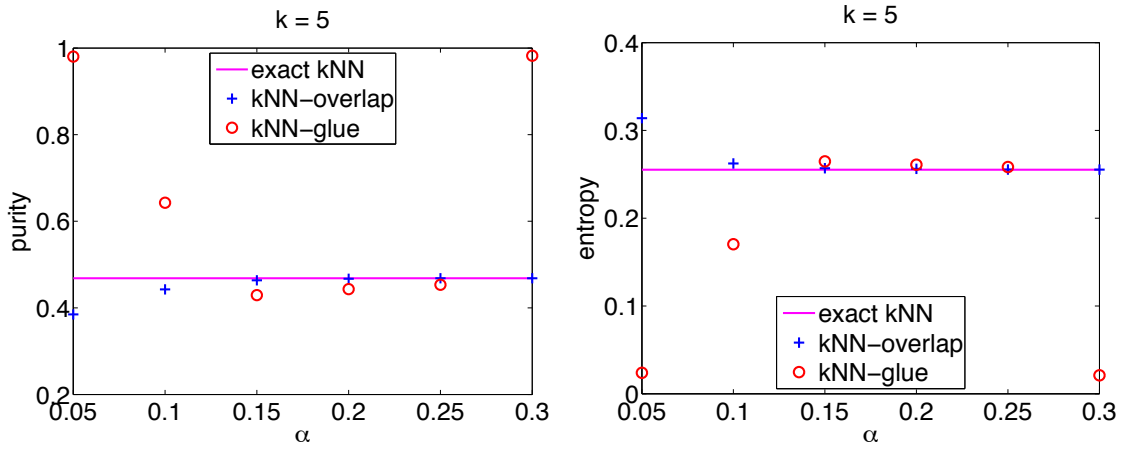
Figure 5: The refine step boosts the accuracy of the graph at some additional computational cost. Data set: MNIST. The settings are the same as in Figure 4.

5.1 Agglomerative Clustering

Agglomerative clustering (Ward, 1963) is a clustering method which exploits a hierarchy for a set of data points. Initially each point belongs to a separate cluster. Then, the algorithm iteratively merges a pair of clusters that has the lowest merge cost, and updates the merge costs between the newly merged cluster and the rest of the clusters. This process terminates when the desired number of clusters is reached (or when all the points belong to the single final cluster). A straightforward implementation of the method takes $O(n^3)$ time, since there are $O(n)$ iterations, each of which takes $O(n^2)$ time to find the pair with the lowest merge cost (Shanbehzadeh and Ogunbona, 1997). Fränti et al. (2006) proposed, at each iteration, to maintain the kNN graph of the clusters, and merge two clusters that are the nearest neighbors in the graph. Let the number of clusters at the present iteration be m. Then this method takes $O(km)$ time to find the closest clusters, compared with the $O(m^2)$ cost of finding the pair with the lowest merge cost. With a delicate implementation using a doubly linked list, they showed that the overall running time of the clustering process reduces to $O(\tau n \log n)$, where τ is the number of nearest neighbor updates at each iteration. Their method greatly speeds up the clustering process, while the clustering quality is not much degraded.

However, the quadratic time to create the initial kNN graph eclipses the improvement in the clustering time. One solution is to use an approximate kNN graph that can be inexpensively created. Virmajoki and Fränti (2004) proposed a divide-and-conquer algorithm to create an approximate kNN graph, but their time complexity was overestimated. Our approach also follows the common framework of divide and conquer. However, we bring three improvements over previous work: (1) Two methods to perform the divide step are proposed, (2) an efficient way to compute the separating hyperplane is described, and (3) a detailed and rigorous analysis on the time complexity is provided. This analysis in particular makes the proposed algorithms practical, especially in the presence of high dimensional data (e.g., when d is in the order of hundreds or thousands).

We performed an experiment on the data set PIE with 68 classes (see Figure 6). Since class labels are known, we used the purity and the entropy (Zhao and Karypis, 2004) as quality measures.



(a) Purity.

(b) Entropy.

(c) Sum of squared errors.

Figure 6: Agglomerative clustering using kNN graphs on the image data set PIE (68 human subjects).

They are defined as

$$\text{Purity} = \sum_{i=1}^{q} \frac{n_i}{n} \text{Purity}(i), \qquad \text{where} \qquad \text{Purity}(i) = \frac{1}{n_i} \max_j \left(n_i^j \right),$$

and

$$\text{Entropy} = \sum_{i=1}^{q} \frac{n_i}{n} \text{Entropy}(i), \qquad \text{where} \qquad \text{Entropy}(i) = -\sum_{j=1}^{q} \frac{n_i^j}{n_i} \log_q \frac{n_i^j}{n_i}.$$

Here, q is the number of classes/clusters, n_i is the size of cluster i, and n_i^j is the number of class j data that are assigned to the i-th cluster. The purity and the entropy both range from 0 to 1. In Figure 6 we show the purities, the entropies, and the values of the objective function for a general purpose clustering (sum of squared errors, SSE), for different methods and different α's. In general,

a higher purity, a lower entropy, and/or a lower SSE means a better clustering quality. As can be seen the qualities of the clusterings obtained from the approximate kNN graphs are very close to those resulting from the exact graph, with a few being even much better. It is interesting to note that the clustering results seem to have little correlation with the qualities of the graphs governed by the value α.

5.2 Dimensionality Reduction

Many dimensionality reduction methods, for example, locally linear embedding (LLE) (Roweis and Saul, 2000), Laplacian eigenmaps (Belkin and Niyogi, 2003), locality preserving projections (LPP) (He and Niyogi, 2004), and orthogonal neighborhood preserving projections (ONPP) (Kokiopoulou and Saad, 2007), compute a low dimensional embedding of the data by preserving the local neighborhoods for each point. For example, in LLE, a weighted adjacency matrix W is first computed to minimize the following objective:

$$\mathcal{E}(W) = \sum_i \left\| x_i - \sum_{x_j \in N(x_i)} W_{ij} x_j \right\|^2, \qquad \text{subject to } \sum_j W_{ij} = 1,\ \forall i,$$

where $N(\cdot)$ means the neighborhood of a point. Then, a low dimensional embedding $Y = [y_1, \ldots, y_n]$ is computed such that it minimizes the objective:

$$\Phi(Y) = \sum_i \left\| y_i - \sum_{y_j \in N(y_i)} W_{ij} y_j \right\|^2, \qquad \text{subject to } YY^T = I.$$

The final solution, Y, is the matrix whose column-vectors are the r bottom right singular vectors of the Laplacian matrix $L_{\text{LLE}} = I - W$. As another example, in Laplacian eigenmaps, the low dimensional embedding $Y = [y_1, \ldots, y_n]$ is computed so as to minimize the cost function:

$$\Psi(Y) = \sum_{i,j} W_{ij} \left\| y_i - y_j \right\|^2, \qquad \text{subject to } YDY^T = I,$$

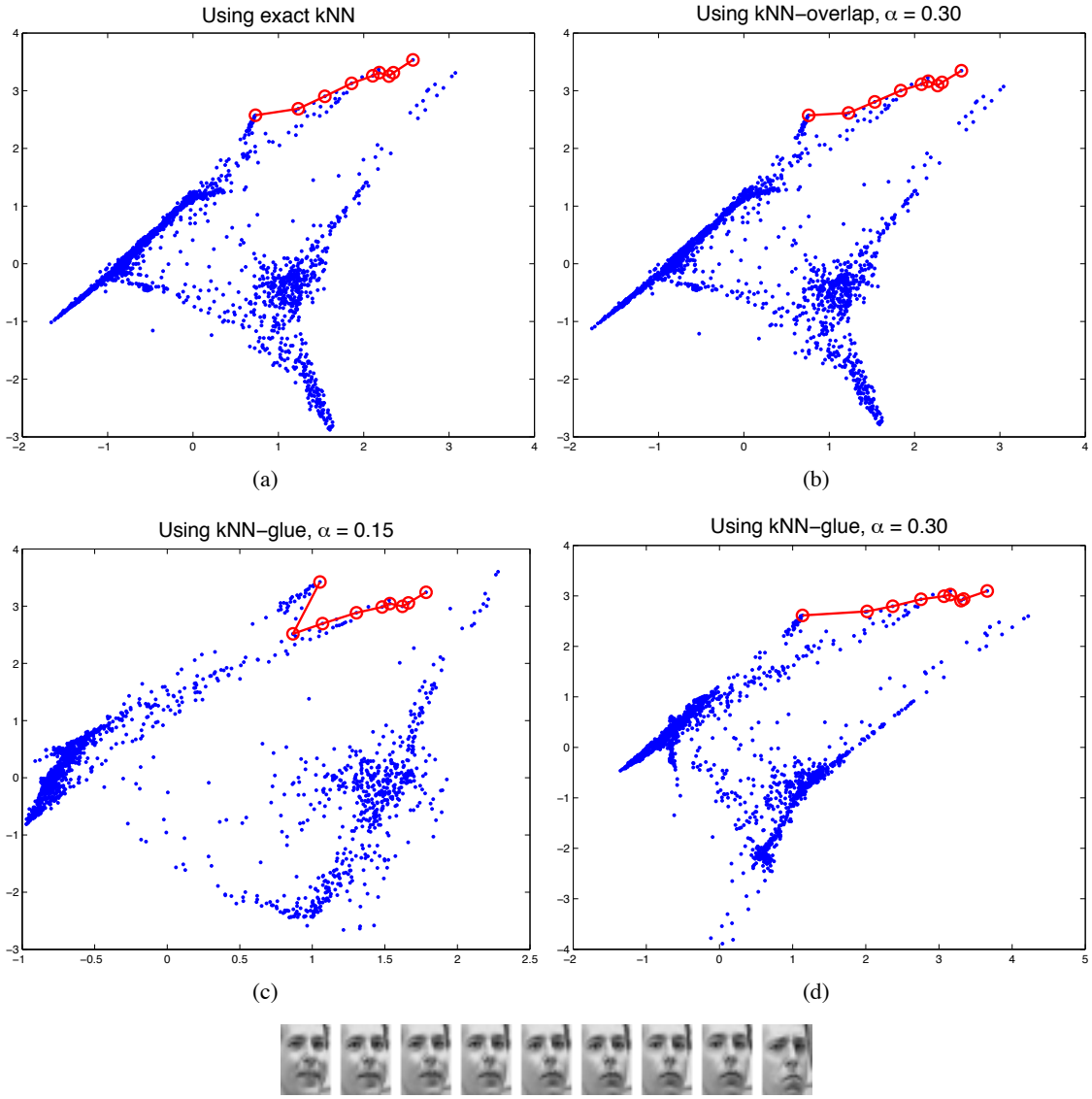
where D is the diagonal degree matrix. Here, the weighted adjacency matrix W is defined in a number of ways, one of the most popular being the weights of the heat kernel

$$W_{ij} = \begin{cases} \exp(-\left\| x_i - x_j \right\|^2 / \sigma^2) & \text{if } x_i \in N(x_j) \text{ or } x_j \in N(x_i), \\ 0 & \text{otherwise.} \end{cases}$$

The solution, Y, is simply the matrix of r bottom eigenvectors of the normalized Laplacian matrix $L_{\text{eigenmaps}} = I - D^{-1/2} W D^{-1/2}$ (subject to scaling).

A thorn in the nicely motivated formulations for the above approaches, is that they all begin with a rather expensive computation to obtain the neighborhood graph of the data. On the other hand, the cost of computing the solution Y via, for example, the Lanczos algorithm, is relatively inexpensive, and can be summarized as[7] $O(rkn)$, which is independent of the dimension d. The methods discussed in this paper are suitable alternatives to the expensive brute-force approach to

7. To be more exact, the dominant cost of computing r singular vectors/eigenvectors of a sparse matrix by the Lanczos method is $O(r' \cdot \text{nnz})$, where r' is the number of Lanczos steps and nnz is the number of nonzeros in the matrix. The value r' is in practice a few times of r, and in our situation, the matrix is the (normalized) graph Laplacian, hence $\text{nnz} = O(kn)$.



(e) Images on the circle path. Facial expressions gradually change.

Figure 7: Dimensionality reduction on the data set FREY by LLE.

extract the exact *k*NN graph, since the approximate graphs are accurate enough for the purpose of dimensionality reduction, while the time costs are significantly smaller. Figures 7 and 8 provide two illustrations of this.

In Figure 7 are the plots of the dimensionality reduction results of LLE applied to the data set FREY, where we used $k = 12$ as that in Roweis and Saul (2000, Figure 3). Figure 7(a) shows the result when using the exact *k*NN graph, while Figure 7(b) shows the result when using the approximate *k*NN graph by the overlap method with $\alpha = 0.30$. It is clear that the two results are almost identical. Figures 7(c) and 7(d) give two plots when the glue method is used. Although the

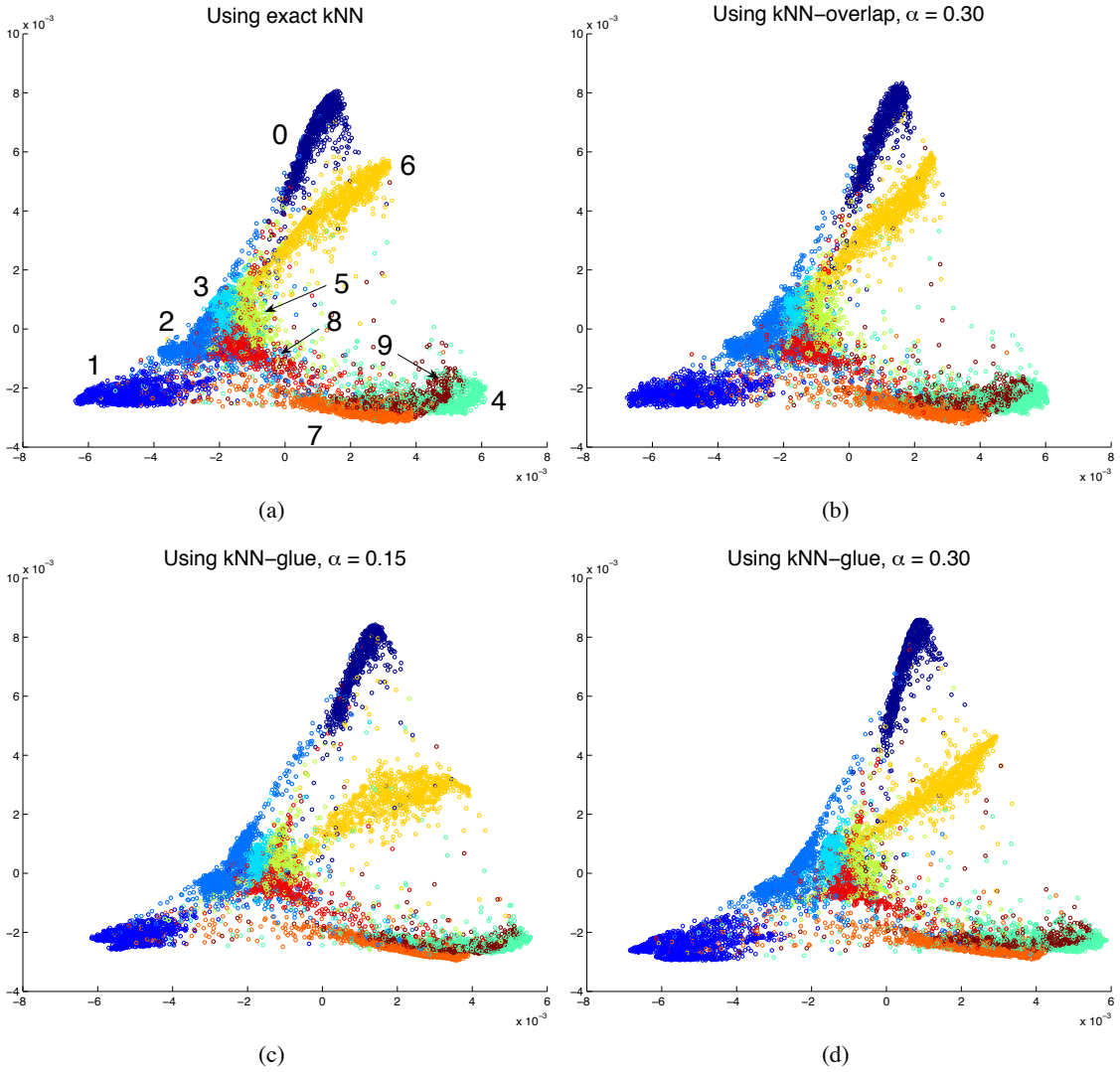


Figure 8: Dimensionality reduction on the data set MNIST by Laplacian eigenmaps.

embeddings are different from those of the exact kNN graph in Figure 7(a), they also represent the original data manifold quite well. This can be seen by tracking the relative locations of the sequence of images (as shown in 7(e)) in the two dimensional space.

Figure 8 shows the plots of the dimensionality reduction results of Laplacian eigenmaps applied to the data set MNIST, where we used $k = 5$. Figure 8(a) shows the original result by using the exact kNN graph, while 8(b), 8(c) and 8(d) show the results by using the overlap method with $\alpha = 0.30$, the glue method with $\alpha = 0.15$, and the glue method with $\alpha = 0.30$, respectively. The four plots all show clear clusterings of the ten classes (digits from 0 to 9), and the localization patterns of the clusterings are very similar.

6. Conclusions

We have proposed two sub-quadratic time methods under the framework of divide and conquer for computing approximate kNN graphs for high dimensional data. The running times of the methods, as well as the qualities of the resulting graphs, depend on an internal parameter that controls the overlap size of the subsets in the divide step. Experiments show that in order to obtain a high quality graph, a small overlap size is usually sufficient and this leads to a small exponent in the time complexity. An avenue of future research is to theoretically analyze the quality of the resulting graphs in relation to the overlap size. The resulting approximate graphs have a wide range of applications as they can be safely used as alternatives to the exact kNN graph. We have shown two such examples: one in agglomerative clustering and the other in dimensionality reduction. Thus, replacing the exact kNN graph construction with one produced by the methods proposed here, can significantly alleviate what currently constitutes a major bottleneck in these applications.

Acknowledgments

This research was supported by the National Science Foundation (grant DMS-0810938) and by the Minnesota Supercomputer Institute. The authors would like to thank the anonymous referees who made many valuable suggestions.

Appendix A. Pseudocodes of Procedures

function $G = k$NN-BRUTEFORCE(X, k)
  for $i \leftarrow 1, \ldots, n$ **do**
    for $j \leftarrow i+1, \ldots, n$ **do**
      Compute $\rho(x_i, x_j) = \left\| x_i - x_j \right\|$
      $\rho(x_j, x_i) = \rho(x_i, x_j)$
    end for
    Set $N(x_i) = \{x_j \mid \rho(x_i, x_j)$ is among the k smallest elements for all $j \neq i\}$
  end for
end function

function $(X_1, X_2) =$ DIVIDE-OVERLAP(X, α)
  Compute the largest right singular vector v of $\hat{X} = X - ce^T$
  Let $V = \{|v_i| \mid i = 1, 2, \ldots, n\}$
  Find $h_\alpha(V)$ ▷ See Section 2.2.1 for the definition
  Set $X_1 = \{x_i \mid v_i \geq -h_\alpha(V)\}$
  Set $X_2 = \{x_i \mid v_i < h_\alpha(V)\}$
end function

function $(X_1, X_2, X_3) =$ DIVIDE-GLUE(X, α)
  Compute the largest right singular vector v of $\hat{X} = X - ce^T$
  Let $V = \{|v_i| \mid i = 1, 2, \ldots, n\}$
  Find $h_\alpha(V)$
  Set $X_1 = \{x_i \mid v_i \geq 0\}$
  Set $X_2 = \{x_i \mid v_i < 0\}$
  Set $X_3 = \{x_i \mid -h_\alpha(V) \leq v_i < h_\alpha(V)\}$
end function

function $G =$ CONQUER(G_1, G_2, k)
  $G = G_1 \cup G_2$
  for all $x \in V(G_1) \cap V(G_2)$ **do** ▷ $V(\cdot)$ denotes the vertex set of the graph
    Update $N(x) \leftarrow \{y \mid \rho(x,y)$ is among the k smallest elements for all $y \in N(x)\}$
  end for
end function

function $G =$ CONQUER(G_1, G_2, G_3, k)
  $G = G_1 \cup G_2 \cup G_3$
  for all $x \in V(G_3)$ **do**
    Update $N(x) \leftarrow \{y \mid \rho(x,y)$ is among the k smallest elements for all $y \in N(x)\}$
  end for
end function

function REFINE(G, k)
  for all $x \in V(G)$ **do**
    Update $N(x) \leftarrow \{y \mid \rho(x,y)$ is among the k smallest elements for all
      $y \in N(x) \cup \left(\bigcup_{z \in N(x)} N(z)\right)\}$
  end for
end function

Appendix B. Proofs

Theorem 1 follows from the Master Theorem (Cormen et al., 2001, Chapter 4.3). Theorem 2 is an immediate consequence of the following lemma, which is straightforward to verify.

Lemma 4 *The recurrence relation*

$$T(n) = 2T(n/2) + T(\alpha n) + n$$

with $T(1) = 1$ *has a solution*

$$T(n) = \left(1 + \frac{1}{\alpha}\right) n^t - \frac{n}{\alpha}$$

where t *is the solution to the equation*

$$\frac{2}{2^t} + \alpha^t = 1.$$

The proof of Theorem 3 requires two lemmas.

Lemma 5 *When* $0 < x < 1$,

$$\log_2(1-x^2) > \Big(\log_2(1-x)\Big)\Big(\log_2(1+x)\Big).$$

Proof By Taylor expansion,

$$\begin{aligned}
&\Big(\ln(1-x)\Big)\Big(\ln(1+x)\Big)\\
&= \left(-\sum_{n=1}^{\infty}\frac{x^n}{n}\right)\left(\sum_{n=1}^{\infty}\frac{(-1)^{n+1}x^n}{n}\right)\\
&= -\sum_{n=1}^{\infty}\left(\sum_{m=1}^{2n-1}\frac{(-1)^{m-1}}{m(2n-m)}\right)x^{2n}\\
&= -\sum_{n=1}^{\infty}\left(\frac{1}{2n}\left(\frac{1}{1}+\frac{1}{2n-1}\right)-\frac{1}{2n}\left(\frac{1}{2}+\frac{1}{2n-2}\right)+\cdots+\frac{(-1)^{n-1}}{n^2}\right.\\
&\qquad\left.+\cdots-\frac{1}{2n}\left(\frac{1}{2n-2}+\frac{1}{2}\right)+\frac{1}{2n}\left(\frac{1}{2n-1}+\frac{1}{1}\right)\right)x^{2n}\\
&= -\sum_{n=1}^{\infty}\left(\frac{1}{n}\left(1-\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{2n-1}\right)\right)x^{2n}\\
&< -\sum_{n=1}^{\infty}\left(\frac{\ln 2}{n}\right)x^{2n}\\
&= (\ln 2)\cdot\ln(1-x^2).
\end{aligned}$$

The inequality in the lemma follows by changing the bases of the logarithms. ■

Lemma 6 *The following inequality*

$$\log_2(ab) > (\log_2 a)(\log_2 b)$$

holds whenever $0 < a < 1 < b < 2$ *and* $a+b \geq 2$.

Proof By using $b \geq 2-a$, we have the following two inequalities

$$\log_2(ab) \geq \log_2(a(2-a)) = \log_2(1-(1-a))(1+(1-a)) = \log_2(1-(1-a)^2)$$

and

$$(\log_2 a)(\log_2 b) \leq (\log_2 a)(\log_2(2-a)) = \log_2(1-(1-a))\times\log_2(1+(1-a)).$$

Then by applying Lemma 5 with $1-a=x$, we have

$$\log_2(1-(1-a)^2) > \log_2(1-(1-a))\times\log_2(1+(1-a)).$$

Thus, the inequality of the lemma holds. ■

Proof of Theorem 3 From Theorem 2 we have

$$t_{\mathrm{g}} = 1 - \log_2(1-\alpha^{t_{\mathrm{g}}}) > 1.$$

Then

$$\begin{aligned} t_{\mathrm{o}} - t_{\mathrm{g}} &= \frac{1}{1-\log_2(1+\alpha)} - 1 + \log_2(1-\alpha^{t_{\mathrm{g}}}) \\ &= \frac{\log_2(1+\alpha) + \log_2(1-\alpha^{t_{\mathrm{g}}}) - \log_2(1+\alpha)\times\log_2(1-\alpha^{t_{\mathrm{g}}})}{1-\log_2(1+\alpha)}. \end{aligned}$$

Since $0 < \alpha < 1$, the denominator $1-\log_2(1+\alpha)$ is positive. By Lemma 6 the numerator is also positive. Hence $t_{\mathrm{o}} > t_{\mathrm{g}}$. ■

References

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computatioin*, 16(6):1373–1396, 2003.

J. Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23:214–229, 1980.

J. Bentley, D. Stanat, and E. Williams. The complexity of finding fixed-radius near neighbors. *Information Processing Letters*, 6:209–213, 1977.

M. W. Berry. Large scale sparse singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49, 1992.

D. L. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4): 324–344, 1998.

M. Brito, E. Chávez, A. Quiroz, and J. Yukich. Connectivity of the mutual *k*-nearest neighbor graph in clustering and outlier detection. *Statistics & Probability Letters*, 35:33–42, 1997.

P. B. Callahan. Optimal parallel all-nearest-neighbors using the well-separated pair decomposition. In *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, 1993.

P. B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to *k*-nearest-neighbors and *n*-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.

B. Chazelle. An improved algorithm for the fixed-radius neighbor problem. *Information Processing Letters*, 16:193–198, 1983.

H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, 2005.

K. L. Clarkson. Fast algorithms for the all nearest neighbors problem. In *Proceedings of the 24th Annual IEEE Symposium on the Foundations of Computer Science*, pages 226–232, 1983.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

B. V. Dasarathy. Nearest-neighbor approaches. In Willi Klosgen, Jan M. Zytkow, and Jan Zyt, editors, *Handbook of Data Mining and Knowledge Discovery*, pages 88–298. Oxford University Press, 2002.

P. Fränti, O. Virmajoki, and V. Hautamäki. Fast agglomerative clustering using a *k*-nearest neighbor graph. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 28(11):1875–1881, 2006.

X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems 16 (NIPS 2004)*, 2004.

P. Indyk. Nearest neighbors in high-dimensional spaces. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*. CRC Press LLC, 2nd edition, 2004.

P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, 1998.

F. Juhász and K. Mályusz. *Problems of cluster analysis from the viewpoint of numerical analysis*, volume 22 of *Colloquia Mathematica Societatis Janos Bolyai*. North-Holland, Amsterdam, 1980.

J. M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, 1997.

E. Kokiopoulou and Y. Saad. Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 29(12):2143–2156, 2007.

E. Kushilevitza, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.

C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45:255–282, 1950.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

K.C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(5):684–698, 2005.

T. Liu, A. W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In *Proceedings of Neural Information Processing Systems (NIPS 2004)*, 2004.

R. Paredes, E. Chávez, K. Figueroa, and G. Navarro. Practical construction of *k*-nearest neighbor graphs in metric spaces. In *Proceedings of the 5th Workshop on Efficient and Experimental Algorithms (WEA'06)*, 2006.

E. Plaku and L. E. Kavraki. Distributed computation of the *k*nn graph for large high-dimensional point sets. *Journal of Parallel and Distributed Computing*, 67(3):346–359, 2007.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

Y. Saad. On the rates of convergence of the Lanczos and the block-Lanczos methods. *SIAM Journal on Numerical Analysis*, 17(5):687–706, 1980.

J. Sankaranarayanan, H. Samet, and A. Varshney. A fast all nearest neighbor algorithm for applications involving large point-clouds. *Computers and Graphics*, 31(2):157–174, 2007.

L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

G. Shakhnarovich, T. Darrell, and P. Indyk, editors. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. The MIT Press, 2006.

J. Shanbehzadeh and P. O. Ogunbona. On the computational complexity of the LBG and PNN algorithms. *IEEE Transactions on Image Processing*, 6(4):614–616, 1997.

T. Sim, S. Baker, and M. Bsat. The CMU pose, illumination, and expression database. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 25(12):1615–1618, 2003.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

D. Tritchler, S. Fallah, and J. Beyene. A spectral clustering method for microarray data. *Computational Statistics & Data Analysis*, 49:63–76, 2005.

P. M. Vaidya. An $O(n \log n)$ algorithm for the all-nearest-neighbors problem. *Discrete Computational Geometry*, 4:101–115, 1989.

O. Virmajoki and P. Fränti. Divide-and-conquer algorithm for creating neighborhood graph for clustering. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)*, 2004.

J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, 2004.

Journal of Machine Learning Research 10 (2009) 2013-2038 Submitted 12/08; Revised 5/09; Published 9/09

Ultrahigh Dimensional Feature Selection: Beyond The Linear Model

Jianqing Fan JQFAN@PRINCETON.EDU
Department of Operations Research and Financial Engineering,
Princeton University,
Princeton, NJ 08540 USA

Richard Samworth R.SAMWORTH@STATSLAB.CAM.AC.UK
Statistical Laboratory,
University of Cambridge,
Cambridge, CB3 0WB, UK

Yichao Wu WU@STAT.NCSU.EDU
Department of Statistics
North Carolina State Univesity
Raleigh, NC 27695 USA

Editor: Saharon Rosset

Abstract

Variable selection in high-dimensional space characterizes many contemporary problems in scientific discovery and decision making. Many frequently-used techniques are based on independence screening; examples include correlation ranking (Fan & Lv, 2008) or feature selection using a two-sample t-test in high-dimensional classification (Tibshirani et al., 2003). Within the context of the linear model, Fan & Lv (2008) showed that this simple correlation ranking possesses a sure independence screening property under certain conditions and that its revision, called iteratively sure independent screening (ISIS), is needed when the features are marginally unrelated but jointly related to the response variable. In this paper, we extend ISIS, without explicit definition of residuals, to a general pseudo-likelihood framework, which includes generalized linear models as a special case. Even in the least-squares setting, the new method improves ISIS by allowing feature deletion in the iterative process. Our technique allows us to select important features in high-dimensional classification where the popularly used two-sample t-method fails. A new technique is introduced to reduce the false selection rate in the feature screening stage. Several simulated and two real data examples are presented to illustrate the methodology.

Keywords: classification, feature screening, generalized linear models, robust regression, feature selection

1. Introduction

The remarkable development of computing power and other technology has allowed scientists to collect data of unprecedented size and complexity. Examples include data from microarrays, proteomics, brain images, videos, functional data and high-frequency financial data. Such a demand from applications presents many new challenges as well as opportunities for those in statistics and machine learning, and while some significant progress has been made in recent years, there remains a great deal to do.

A very common statistical problem is to model the relationship between one or more output variables Y and their associated covariates (or features) $X_1, \ldots, X_p$, based on a sample of size n. A characteristic feature of many of the modern problems mentioned in the previous paragraph is that the dimensionality p is large, potentially much larger than n. Mathematically, it makes sense to consider p as a function of n, which diverges to infinity. The dimensionality grows very rapidly when interactions of the features are considered, which is necessary for many scientific endeavors. For example, in disease classification using microarray gene expression data (Tibshirani et al., 2003; Fan & Ren, 2006), the number of arrays is usually in the order of tens or hundreds while the number of gene expression profiles is in the order of tens of thousands; in the study of protein-protein interactions, the sample size may be in the order of thousands, but the number of features can be in the order of millions.

The phenomenon of noise accumulation in high-dimensional classification and regression has long been observed by statisticians and computer scientists (see Vapnik 1995, Hastie et al. 2009 and references therein) and has been analytically demonstrated by Fan & Fan (2008). Various feature selection techniques have been proposed in both the statistics and machine learning literature, and introductions and overviews written for the machine learning community can be found in, for example, Liu & Motoda (1998), Guyon & Elisseeff (2003) and Guyon et al. (2006). Specific algorithms proposed include but are not restricted to FCBF (Yu & Li, 2003), CFS (Hall, 2000), ReliefF (Kononenko, 1994), FOCUS (Almuallim & Dietterich, 1994) and INTERACT (Zhao & Liu, 2007). See also the special issue published by JMLR on "variable and feature selection", including Bi et al. (2003), Bengio & Chapados (2003) and Guyon & Elisseeff (2003).

One particularly popular family of methods is based on penalized least-squares or, more generally, penalized pseudo-likelihood. Examples include the LASSO (Tibshirani, 1996), SCAD (Fan & Li, 2001), the Dantzig selector (Candes & Tao, 2007), and their related methods. These methods have attracted a great deal of theoretical study and algorithmic development recently. See Donoho & Elad (2003), Efron et al. (2004), Zou (2006), Meinshausen & Bühlmann (2006), Zhao & Yu (2006), Zou & Li (2008), Bickel et al. (2009), and references therein. However, computation inherent in those methods makes them hard to apply directly to ultrahigh-dimensional statistical learning problems, which involve the simultaneous challenges of computational expediency, statistical accuracy, and algorithmic stability.

A method that takes up the aforementioned three challenges is the idea of independent learning, proposed and demonstrated by Fan & Lv (2008) in the regression context. The method can be derived from an empirical likelihood point of view (Hall et al., 2009) and is related to supervised principal component analysis (Bair et al., 2006; Paul et al., 2008). In the important, but limited, context of the linear model, Fan & Lv (2008) proposed a two-stage procedure to deal with this problem. First, so-called independence screening is used as a fast but crude method of reducing the dimensionality to a more moderate size (usually below the sample size); then, a more sophisticated technique, such as a penalized likelihood method based on the smoothly clipped absolute deviation (SCAD) penalty, can be applied to perform the final feature selection and parameter estimation simultaneously.

Independence screening recruits those features having the best marginal utility, which corresponds to the largest marginal correlation with the response in the context of least-squares regression. Under certain regularity conditions, Fan & Lv (2008) show surprisingly that this fast feature selection method has a 'sure screening property'; that is, with probability very close to 1, the independence screening technique retains all of the important features in the model. A remarkable

feature of this theoretical result is that the dimensionality of the model is allowed to grow exponentially in the sample size; for this reason, we refer to the method as an 'ultrahigh' dimensional feature selection technique, to distinguish it from other 'high' dimensional methods where the dimensionality can grow only polynomially in the sample size. The sure screening property is described in greater detail in Section 3.2, and as a result of this theoretical justification, the method is referred to as Sure Independence Screening (SIS).

An important methodological extension, called Iterated Sure Independence Screening (ISIS), covers cases where the regularity conditions may fail, for instance if a feature is marginally uncorrelated, but jointly correlated with the response, or the reverse situation where a feature is jointly uncorrelated but has higher marginal correlation than some important features. Roughly, ISIS works by iteratively performing feature selection to recruit a small number of features, computing residuals based on the model fitted using these recruited features, and then using the working residuals as the response variable to continue recruiting new features. The crucial step is to compute the working residuals, which is easy for the least-squares regression problem but not obvious for other problems. The improved performance of ISIS has been documented in Fan & Lv (2008).

Independence screening is a commonly used techniques for feature selection. It has been widely used for gene selection or disease classification in bioinformatics. In those applications, the genes or proteins are called statistically significant if their associated expressions in the treatment group differ statistically from the control group, resulting in a large and active literature on the multiple testing problem. See, for example, Dudoit et al. (2003) and Efron (2008). The selected features are frequently used for tumor/disease classification. See, for example, Tibshirani et al. (2003), and Fan & Ren (2006). This screening method is indeed a form of independence screening and has been justified by Fan & Fan (2008) under some ideal situations. However, common sense can carry us only so far. As indicated above and illustrated further in Section 4.1, it is easy to construct features that are marginally unrelated, but jointly related with the response. Such features will be screened out by independent learning methods such as the two-sample t test. In other words, genes that are screened out by test statistics can indeed be important in disease classification and understanding molecular mechanisms of the disease. How can we construct better feature selection procedures in ultrahigh dimensional feature space than the independence screening popularly used in feature selection?

The first goal of this paper is to extend SIS and ISIS to much more general models. One challenge here is to make an appropriate definition of a residual in this context. We describe a procedure that effectively sidesteps this issue and therefore permits the desired extension of ISIS. In fact, our method even improves the original ISIS of Fan & Lv (2008) in that it allows variable deletion in the recruiting process. Our methodology applies to a very general pseudo-likelihood framework, in which the aim is to find the parameter vector $\beta = (\beta_1, \ldots, \beta_p)^T$ that is sparse and minimizes an objective function of the form

$$Q(\beta_0, \beta) = \sum_{i=1}^{n} L(Y_i, \beta_0 + \mathbf{x}_i^T \beta),$$

where $(\mathbf{x}_i^T, Y_i)$ are the covariate vector and response for the i^{th} individual. Important applications of this methodology, which is outlined in greater detail in Section 2, include the following:

1. **Generalized linear models**: All generalized linear models, including logistic regression and Poisson log-linear models, fit very naturally into our methodological framework. See McCullagh & Nelder (1989) for many applications of generalized linear models. Note in particular

that logistic regression models yield a popular approach for studying classification problems. In Section 4, we present simulations in which our approach compares favorably with the competing LASSO technique (Tibshirani, 1996).

2. **Classification**: Other common approaches to classification assume the response takes values in $\{-1,1\}$ and also fit into our framework. For instance, support vector machine classifiers (Vapnik, 1995) use the hinge loss function $L(Y_i, \beta_0 + \mathbf{x}_i^T\beta) = \{1 - Y_i(\beta_0 + \mathbf{x}_i^T\beta)\}_+$, while the boosting algorithm AdaBoost (Freund & Schapire, 1997) uses $L(Y_i, \beta_0 + \mathbf{x}_i^T\beta) = \exp\{-Y_i(\beta_0 + \mathbf{x}_i^T\beta)\}$.

3. **Robust fitting**: In a high-dimensional linear model setting, it is advisable to be cautious about the assumed relationship between the features and the response. Thus, instead of the conventional least squares loss function, we may prefer a robust loss function such as the L_1 loss $L(Y_i, \beta_0 + \mathbf{x}_i^T\beta) = |Y_i - \beta_0 - \mathbf{x}_i^T\beta|$ or the Huber loss (Huber, 1964), which also fits into our framework.

Any screening method, by default, has a large false selection rate (FSR), namely, many unimportant features are selected after screening. A second aim of this paper, covered in Section 3, is to present two variants of the SIS methodology, which reduce significantly the FSR. Both are based on partitioning the data into (usually) two groups. The first has the desirable property that in high-dimensional problems the probability of incorrectly selecting unimportant features is small. Thus this method is particularly useful as a means of quickly identifying features that should be included in the final model. The second method is less aggressive, and for the linear model has the same sure screening property as the original SIS technique. The applications of our proposed methods are illustrated in Section 5.

2. ISIS Methodology in a General Framework

Let $\mathbf{y} = (Y_1, \ldots, Y_n)^T$ be a vector of responses and let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be their associated covariate (column) vectors, each taking values in $\mathbb{R}^p$. The vectors $(\mathbf{x}_1^T, Y_1), \ldots, (\mathbf{x}_n^T, Y_n)$ are assumed to be independent and identically distributed realizations from the population $(X_1, \ldots, X_p, Y)^T$. The $n \times p$ design matrix is $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$.

2.1 Feature Ranking by Marginal Utilities

The relationship between Y and $(X_1, \ldots, X_p)^T$ is often modeled through a parameter vector $\beta = (\beta_1, \ldots, \beta_p)^T$, and the fitting of the model amounts to minimizing a negative pseudo-likelihood function of the form

$$Q(\beta_0, \beta) = n^{-1} \sum_{i=1}^{n} L(Y_i, \beta_0 + \mathbf{x}_i^T\beta).$$

Here, L can be regarded as the loss of using $\beta_0 + \mathbf{x}_i^T\beta$ to predict Y_i. The marginal utility of the j^{th} feature is

$$L_j = \min_{\beta_0, \beta_j} n^{-1} \sum_{i=1}^{n} L(Y_i, \beta_0 + X_{ij}\beta_j),$$

which minimizes the loss function, where $\mathbf{x}_i = (X_{i1}, \ldots, X_{ip})^T$. The idea of SIS in this framework is to compute the vector of marginal utilities $\mathbf{L} = (L_1, \ldots, L_p)^T$ and rank them according to the

marginal utilities: the smaller the more important. Note that in order to compute L_j, we need only fit a model with two parameters, β_0 and β_j, so computing the vector $\mathbf{L}$ can be done very quickly and stably, even for an ultrahigh dimensional problem. The feature X_j is selected by SIS if L_j is one of the d smallest components of $\mathbf{L}$. Typically, we may take $d = \lfloor n/\log n \rfloor$, though the choice of d is discussed in greater detail in Section 4.

The procedure above is an independence screening method. It uses only a marginal relation between features and the response variable to screen variables. When d is large enough, it has high probability of selecting all of the important features. For this reason, we call the method *Sure Independence Screening* (SIS). For classification problems with quadratic loss L, Fan & Lv (2008) show that SIS reduces to feature screening using a two-sample t-statistic. See also Hall et al. (2009) for a derivation from an empirical likelihood point of view and §3.2 for some theoretical results on the sure screening property.

2.2 Penalized Pseudo-likelihood

With features crudely selected by SIS, variable selection and parameter estimation can further be carried out simultaneously using a more refined penalized (pseudo)-likelihood method, as we now describe. The approach takes joint information into consideration. By reordering the features if necessary, we may assume without loss of generality that $X_1, \ldots, X_d$ are the features recruited by SIS. We let $\mathbf{x}_{i,d} = (X_{i1}, \ldots, X_{id})^T$ and redefine $\beta = (\beta_1, \ldots, \beta_d)^T$. In the penalized likelihood approach, we seek to minimize

$$\ell(\beta_0, \beta) = n^{-1} \sum_{i=1}^{n} L(Y_i, \beta_0 + \mathbf{x}_{i,d}^T \beta) + \sum_{j=1}^{d} p_\lambda(|\beta_j|). \tag{1}$$

Here, $p_\lambda(\cdot)$ is a penalty function and $\lambda > 0$ is a regularization parameter, which may be chosen by five-fold cross-validation, for example. The penalty function should satisfy certain conditions in order for the resulting estimates to have desirable properties, and in particular to yield sparse solutions in which some of the coefficients may be set to zero; see Fan & Li (2001) for further details.

Commonly used examples of penalty functions include the L_1 penalty $p_\lambda(|\beta|) = \lambda|\beta|$ (Tibshirani, 1996; Park & Hastie, 2007), the smoothly clipped absolute deviation (SCAD) penalty (Fan & Li, 2001), which is a quadratic spline with $p_\lambda(0) = 0$ and

$$p'_\lambda(|\beta|) = \lambda \left\{ \mathbb{1}_{\{|\beta| \le \lambda\}} + \frac{(a\lambda - |\beta|)_+}{(a-1)\lambda} \mathbb{1}_{\{|\beta| > \lambda\}} \right\},$$

for some $a > 2$ and $|\beta| > 0$, and the minimum concavity penalty (MCP), $p'_\lambda(|\beta|) = (\lambda - |\beta|/a)_+$ (Zhang, 2009). The choice $a = 3.7$ has been recommended in Fan & Li (2001). Unlike the L_1 penalty, SCAD and MC penalty functions have flat tails, which are fundamental in reducing biases due to penalization (Antoniadis & Fan, 2001; Fan & Li, 2001). Park & Hastie (2007) describe an iterative algorithm for minimizing the objective function for the L_1 penalty, and Zhang (2009) propose a PLUS algorithm for finding solution paths to the penalized least-squares problem with a general penalty $p_\lambda(\cdot)$. On the other hand, Fan & Li (2001) have shown that the SCAD-type of penalized loss function can be minimized iteratively using a local quadratic approximation, whereas Zou & Li (2008) propose a local linear approximation, taking the advantage of recently developed

algorithms for penalized L_1 optimization (Efron et al., 2004). Starting from $\beta^{(0)} = 0$ as suggested by Fan & Lv (2008), using the local linear approximation

$$p_\lambda(|\beta|) \approx p_\lambda(|\beta^{(k)}|) + p'_\lambda(|\beta^{(k)}|)(|\beta| - |\beta^{(k)}|),$$

in (1), at the $(k+1)^{th}$ iteration we minimize the weighted L_1 penalty

$$n^{-1}\sum_{i=1}^{n} L(Y_i, \beta_0 + \mathbf{x}_{i,d}^T\beta) + \sum_{j=1}^{d} w_j^{(k)}|\beta_j|, \tag{2}$$

where $w_j^{(k)} = p'_\lambda(|\beta_j^{(k)}|)$. Note that with initial value $\beta^{(0)} = 0$, $\beta^{(1)}$ is indeed a LASSO estimate for the SCAD and MC penalties, since $p'_\lambda(0+) = \lambda$. In other words, zero is not an absorbing state. Though motivated slightly differently, a weighted L_1 penalty is also the basis of the adaptive Lasso (Zou, 2006); in that case $w_j^{(k)} \equiv w_j = 1/|\hat{\beta}_j|^\gamma$, where $\widehat{\beta} = (\hat{\beta}_1, \ldots, \hat{\beta}_d)^T$ may be taken to be the maximum likelihood estimator, and $\gamma > 0$ is chosen by the user. The drawback of such an approach is that zero is an absorbing state when (2) is iteratively used—components being estimated as zero at one iteration will never escape from zero.

For a class of penalty functions that includes the SCAD penalty and when d is fixed as n diverges, Fan & Li (2001) established an oracle property; that is, the penalized estimates perform asymptotically as well as if an oracle had told us in advance which components of β were non-zero. Fan & Peng (2004) extended this result to cover situations where d may diverge with $d = d_n = o(n^{1/5})$. Zou (2006) shows that the adaptive LASSO possesses the oracle property too, when d is finite. See also further theoretical studies by Zhang & Huang (2008) and Zhang (2009). We refer to the two-stage procedures described above as SIS-Lasso, SIS-SCAD and SIS-AdaLasso.

2.3 Iterative Feature Selection

The SIS methodology may break down if a feature is marginally unrelated, but jointly related with the response, or if a feature is jointly uncorrelated with the response but has higher marginal correlation with the response than some important features. In the former case, the important feature has already been screened at the first stage, whereas in the latter case, the unimportant feature is ranked too high by the independent screening technique. ISIS seeks to overcome these difficulties by using more fully the joint covariate information while retaining computational expedience and stability as in SIS.

In the first step, we apply SIS to pick a set $\widehat{\mathcal{A}}_1$ of indices of size k_1, and then employ a penalized (pseudo)-likelihood method such as Lasso, SCAD, MCP or the adaptive Lasso to select a subset $\widehat{\mathcal{M}}_1$ of these indices. This is our initial estimate of the set of indices of important features. The screening stage solves only bivariate optimizations (2.1) and the fitting part solves only a optimization problem (1) with moderate size k_1. This is an attractive feature in ultrahigh dimensional statistical learning.

Instead of computing residuals, as could be done in the linear model, we compute

$$L_j^{(2)} = \min_{\beta_0, \beta_{\widehat{\mathcal{M}}_1}, \beta_j} n^{-1}\sum_{i=1}^{n} L(Y_i, \beta_0 + \mathbf{x}_{i,\widehat{\mathcal{M}}_1}^T\beta_{\widehat{\mathcal{M}}_1} + X_{ij}\beta_j), \tag{3}$$

for $j \in \widehat{\mathcal{M}}_1^c = \{1, \ldots, p\} \setminus \widehat{\mathcal{M}}_1$, where $\mathbf{x}_{i,\widehat{\mathcal{M}}_1}$ is the sub-vector of $\mathbf{x}_i$ consisting of those elements in $\widehat{\mathcal{M}}_1$. This is again a low-dimensional optimization problem which can easily be solved. Note that

$L_j^{(2)}$ [after subtracting the constant $\min_{\beta_0,\beta_{\widehat{\mathcal{M}}_1}} n^{-1}\sum_{i=1}^n L(Y_i,\beta_0+\mathbf{x}_{i,\widehat{\mathcal{M}}_1}^T\beta_{\widehat{\mathcal{M}}_1})$ and changing the sign of the difference] can be interpreted as the additional contribution of variable X_j given the existence of variables in $\widehat{\mathcal{M}}_1$. After ordering $\{L_j^{(2)} : j \in \widehat{\mathcal{M}}_1^c\}$, we form the set $\widehat{\mathcal{A}}_2$ consisting of the indices corresponding to the smallest k_2 elements, say. In this screening stage, an alternative approach is to substitute the fitted value $\widehat{\beta}_{\widehat{\mathcal{M}}_1}$ from the first stage into (3) and the optimization problem (3) would only be bivariate. This approach is exactly an extension of Fan & Lv (2008) as we have

$$L(Y_i,\beta_0+\mathbf{x}_{i,\widehat{\mathcal{M}}_1}^T\widehat{\beta}_{\widehat{\mathcal{M}}_1}+X_{ij}\beta_j)=(\hat{r}_i-\beta_0-X_{ij}\beta_j)^2,$$

for the quadratic loss, where $\hat{r}_i = Y_i - \mathbf{x}_{i,\widehat{\mathcal{M}}_1}^T\widehat{\beta}_{\widehat{\mathcal{M}}_1}$ is the residual from the previous step of fitting. The conditional contributions of features are more relevant in recruiting variables at the second stage, but the computation is more expensive. Our numerical experiments in Section 4.4 shows the improvement of such a deviation from Fan & Lv (2008).

After the prescreening step, we use penalized likelihood to obtain

$$\widehat{\beta}_2 = \underset{\beta_0,\beta_{\widehat{\mathcal{M}}_1},\beta_{\mathcal{A}_2}}{\operatorname{argmin}} \; n^{-1}\sum_{i=1}^n L(Y_i,\beta_0+\mathbf{x}_{i,\widehat{\mathcal{M}}_1}^T\beta_{\widehat{\mathcal{M}}_1}+\mathbf{x}_{i,\widehat{\mathcal{A}}_2}^T\beta_{\widehat{\mathcal{A}}_2})+\sum_{j\in\widehat{\mathcal{M}}_1\cup\widehat{\mathcal{A}}_2} p_\lambda(|\beta_j|).$$

Again, the penalty term encourages a sparse solution. The indices of $\widehat{\beta}_2$ that are non-zero yield a new estimated set $\widehat{\mathcal{M}}_2$ of active indices. This step also deviates importantly from the approach in Fan & Lv (2008) even in the least-squares case. It allows the procedure to delete variables from the previously selected features with indices in $\widehat{\mathcal{M}}_1$.

The process, which iteratively recruits and deletes features, can then be repeated until we obtain a set of indices $\widehat{\mathcal{M}}_\ell$ which either has reached the prescribed size d, or satisfies $\widehat{\mathcal{M}}_\ell = \widehat{\mathcal{M}}_{\ell-1}$. Of course, we also obtain a final estimated parameter vector $\widehat{\beta}_\ell$. The above method can be considered as an analogue of the least squares ISIS procedure (Fan & Lv, 2008) without explicit definition of the residuals. In fact, it is an improvement even for the least-squares problem.

In general, choosing larger values of each k_r decreases the computational cost and the probability that the ISIS procedure will terminate prematurely. However, it also makes the procedure more like its non-iterated counterpart, and so may offer less improvement in the awkward cases for SIS described in Section 1. In our implementation, we chose $k_1 = \lfloor 2d/3 \rfloor$, and thereafter at the rth iteration, we took $k_r = d - |\widehat{\mathcal{M}}_{r-1}|$. This ensures that the iterated versions of SIS take at least two iterations to terminate; another possibility would be to take, for example, $k_r = \min(5, d - |\widehat{\mathcal{M}}_{r-1}|)$.

Fan & Lv (2008) showed empirically that for the linear model ISIS improves significantly the performance of SIS in the difficult cases described above. The reason is that the fitting of the residuals from the $(r-1)^{th}$ iteration on the remaining features significantly weakens the priority of those unimportant features that are highly correlated with the response through their associations with $\{X_j : j \in \widehat{\mathcal{M}}_{r-1}\}$. This is due to the fact that the features $\{X_j : j \in \widehat{\mathcal{M}}_{r-1}\}$ have lower correlation with the residuals than with the original responses. It also gives those important features that are missed in the previous step a chance to survive.

2.4 Generalized Linear Models

Recall that we say that Y is of exponential dispersion family form if its density can be written in terms of its mean μ and a dispersion parameter ϕ as

$$f_Y(y;\mu,\phi) = \exp\left\{\frac{y\theta(\mu) - b(\theta(\mu))}{\phi} + c(y,\phi)\right\},$$

from some known functions $\theta(\cdot)$, $b(\cdot)$ and $c(\cdot,\cdot)$. In a generalized linear model for independent responses $Y_1,\ldots,Y_n$, we assert that the conditional density of Y_i given the covariate vector $\mathbf{X}_i = \mathbf{x}_i$ is of exponential dispersion family form, with the conditional mean response μ_i related to $\mathbf{x}_i$ through $g(\mu_i) = \mathbf{x}_i^T\beta$ for some known link function $g(\cdot)$, and where the dispersion parameters are constrained by requiring that $\phi_i = \phi a_i$, for some unknown dispersion parameter ϕ and known constants $a_1,\ldots,a_n$. For simplicity, throughout the paper, we take a constant dispersion parameter.

It is immediate from the form of the likelihood function for a generalized linear model that such a model fits within the pseudo-likelihood framework of Section 4. In fact, we have in general that

$$L(Y_i,\beta_0+\mathbf{x}_i^T\beta) = \sum_{i=1}^n \big\{b\big(\theta(g^{-1}(\beta_0+\mathbf{x}_i^T\beta)\big) - Y_i\theta\big(g^{-1}(\beta_0+\mathbf{x}_i^T\beta)\big)\big\}. \tag{4}$$

If we make the canonical choice of link function, $g(\cdot) = \theta(\cdot)$, then (4) simplifies to

$$L(Y_i,\beta_0+\mathbf{x}_i^T\beta) = \sum_{i=1}^n \big\{b(\beta_0+\mathbf{x}_i^T\beta) - Y_i(\beta_0+\mathbf{x}_i^T\beta)\big\}.$$

An elegant way to handle classification problems is to assume the class label takes values 0 or 1, and fit a logistic regression model. For this particular generalized linear model, we have

$$L(Y_i,\beta_0+\mathbf{x}_i^T\beta) = \sum_{i=1}^n \{\log(1+e^{\beta_0+\mathbf{x}_i^T\beta}) - Y_i(\beta_0+\mathbf{x}_i^T\beta)\},$$

while for Poisson log-linear models, we may take

$$L(Y_i,\beta_0+\mathbf{x}_i^T\beta) = \sum_{i=1}^n \big\{e^{\beta_0+\mathbf{x}_i^T\beta} - Y_i(\beta_0+\mathbf{x}_i^T\beta)\big\}.$$

3. Reduction of False Selection Rates

Sure independence screening approaches are simple and quick methods to screen out irrelevant features. They are usually conservative and include many unimportant features. In this section, we outline two possible variants of SIS and ISIS that have attractive theoretical properties in terms of reducing the FSRs. The first is an aggressive feature selection method that is particularly useful when the dimensionality is very large relative to the sample size; the second is a more conservative procedure.

3.1 First Variant of ISIS

It is convenient to introduce some new notation. We write $\mathcal{A}$ for the set of active indices—that is, the set containing those indices j for which $\beta_j \neq 0$ in the true model. Write $X_{\mathcal{A}} = \{X_j : j \in \mathcal{A}\}$ and $X_{\mathcal{A}^c} = \{X_j : j \in \mathcal{A}^c\}$ for the corresponding sets of active and inactive variables respectively.

Assume for simplicity that n is even, and split the sample into two halves at random. Apply SIS or ISIS separately to the data in each partition (with $d = \lfloor n/\log n \rfloor$ or larger, say), yielding two estimates $\widehat{\mathcal{A}}^{(1)}$ and $\widehat{\mathcal{A}}^{(2)}$ of the set of active indices $\mathcal{A}$. Both of them should have large FSRs, as they are constructed from a crude screening method. Assume that both sets have the satisfy

$$P(\mathcal{A} \subset \widehat{\mathcal{A}}^{(j)}) \to 1, \quad \text{for } j = 1 \text{ and } 2.$$

Then, the active features should appear in both sets with probability tending to one. We thus construct $\widehat{\mathcal{A}} = \widehat{\mathcal{A}}^{(1)} \cap \widehat{\mathcal{A}}^{(2)}$ as an estimate of $\mathcal{A}$. This estimate also satisfies

$$P(\mathcal{A} \subset \widehat{\mathcal{A}}) \to 1.$$

However, this estimate contains many fewer indices corresponding to inactive features, as such indices have to appear twice at random in the sets $\widehat{\mathcal{A}}^{(1)}$ and $\widehat{\mathcal{A}}^{(2)}$. This is indeed shown in Theorem 1 below.

Just as in the original formulation of SIS in Section 2, we can now use a penalized (pseudo)-likelihood method such as SCAD to perform final feature selection from $\widehat{\mathcal{A}}$ and parameter estimation. We can even proceed without the penalization since the false selection rate is small.

In our theoretical support for this variant of SIS, we will make use of the following condition:

(A1) Let $r \in \mathbb{N}$, the set of natural numbers. We say the model satisfies the exchangeability condition at level r if the set of random vectors

$$\{(Y, X_{\mathcal{A}}, X_{j_1}, \ldots, X_{j_r}) : j_1, \ldots, j_r \text{ are distinct elements of } \mathcal{A}^c\}$$

is exchangeable.

This condition ensures that each inactive feature is equally likely to be recruited by SIS. Note that (**A1**) certainly allows inactive features to be correlated with the response, but does imply that each inactive feature has the same marginal distribution. In Theorem 1 below, the case $r = 1$ is particularly important, as it gives an upper bound on the probability of recruiting any inactive features into the model. Note that this upper bound requires only the weakest version (level 1) of the exchangeability condition.

Theorem 1 *Let $r \in \mathbb{N}$, and assume the model satisfies the exchangeability condition **(A1)** at level r. If $\widehat{\mathcal{A}}$ denotes the estimator of $\mathcal{A}$ from the above variant of SIS, then*

$$P(|\widehat{\mathcal{A}} \cap \mathcal{A}^c| \geq r) \leq \frac{\binom{d}{r}^2}{\binom{p-|\mathcal{A}|}{r}} \leq \frac{1}{r!}\Big(\frac{d^2}{p-|\mathcal{A}|}\Big)^r,$$

where, for the second inequality, we require $d^2 \leq p - |\mathcal{A}|$ and d is the prescribed number of selected features in $\widehat{\mathcal{A}}^{(1)}$ or $\widehat{\mathcal{A}}^{(2)}$.

Proof Fix $r \in \mathbb{N}$, and let $\mathcal{J} = \{(j_1, \ldots, j_r) : j_1, \ldots, j_r \text{ are distinct elements of } \mathcal{A}^c\}$. Then

$$\begin{aligned} P(|\widehat{\mathcal{A}} \cap \mathcal{A}^c| \geq r) &\leq \sum_{(j_1,\ldots,j_r)\in\mathcal{J}} P(j_1 \in \widehat{\mathcal{A}}, \cdots, j_r \in \widehat{\mathcal{A}}) \\ &= \sum_{(j_1,\ldots,j_r)\in\mathcal{J}} P(j_1 \in \widehat{\mathcal{A}}^{(1)}, \cdots, j_r \in \widehat{\mathcal{A}}^{(1)})^2, \end{aligned}$$

in which we use the random splitting in the last equality. Obviously, the last probability is bounded by

$$\max_{(j_1,\ldots,j_r)\in\mathcal{J}} P(j_1 \in \widehat{\mathcal{A}}^{(1)},\cdots,j_r \in \widehat{\mathcal{A}}^{(1)}) \sum_{(j_1,\ldots,j_r)\in\mathcal{J}} P(j_1 \in \widehat{\mathcal{A}}^{(1)},\cdots,j_r \in \widehat{\mathcal{A}}^{(1)}). \tag{5}$$

Since there are at most d inactive features from $\mathcal{A}^c$ in the set $\widehat{\mathcal{A}}^{(1)}$, the number of r-tuples from $\mathcal{J}$ falling in the set $\widehat{\mathcal{A}}^{(1)}$ can not be more than the total number of such r-tuples in $\widehat{\mathcal{A}}^{(1)}$, that is,

$$\sum_{(j_1,\ldots,j_r)\in\mathcal{J}} \mathbb{1}_{\{j_1\in\widehat{\mathcal{A}}^{(1)},\cdots,j_r\in\widehat{\mathcal{A}}^{(1)}\}} \leq \binom{d}{r}.$$

Thus, we have

$$\sum_{(j_1,\ldots,j_r)\in\mathcal{J}} P(j_1 \in \widehat{\mathcal{A}}^{(1)},\cdots,j_r \in \widehat{\mathcal{A}}^{(1)}) \leq \binom{d}{r}. \tag{6}$$

Substituting this into (5), we obtain

$$P(|\widehat{\mathcal{A}}\cap\mathcal{A}^c| \geq r) \leq \binom{d}{r} \max_{(j_1,\ldots,j_r)\in\mathcal{J}} P(j_1 \in \widehat{\mathcal{A}}^{(1)},\cdots,j_r \in \widehat{\mathcal{A}}^{(1)}).$$

Now, under the exchangeability condition (**A1**), each r-tuple of distinct indices in $\mathcal{A}^c$ is equally likely to be recruited into $\widehat{\mathcal{A}}^{(1)}$. Hence, it follows from (6) that

$$\max_{(j_1,\ldots,j_r)\in\mathcal{J}} P(j_1 \in \widehat{\mathcal{A}}^{(1)},\cdots,j_r \in \widehat{\mathcal{A}}^{(1)}) \leq \frac{\binom{d}{r}}{\binom{p-|\mathcal{A}|}{r}},$$

and the first result follows. The second result follows from the simple fact that

$$\frac{(d-i)^2}{p^*-i} \leq \frac{d^2}{p^*}, \quad \text{for all } 0 \leq i \leq d,$$

where $p^* = p - |\mathcal{A}|$, and the simple calculation that

$$\frac{\binom{d}{r}^2}{\binom{p^*}{r}} = \frac{1}{r!}\frac{d^2(d-1)^2\cdots(d-r+1)^2}{p^*(p^*-1)\cdots(p^*-r+1)} \leq \frac{1}{r!}\left(\frac{d}{p^*}\right)^r.$$

This completes the proof. ■

Theorem 1 gives a nonasymptotic bound, using only the symmetry arguments, and this bound is expected to be reasonably tight especially when p is large. From Theorem 1, we see that if the exchangeability condition at level 1 is satisfied and if p is large by comparison with n^2, then when the number of selected features d is smaller than n, we have with high probability this variant of SIS reports no 'false positives'; that is, it is very likely that any index in the estimated active set also belongs to the active set in the true model. Intuitively, if p is large, then each inactive feature has small probability of being included in the estimated active set, so it is very unlikely indeed that it will appear in the estimated active sets from both partitions. The nature of this result is a little unusual in that it suggests a 'blessing of dimensionality'—the bound on the probability of false

positives decreases with p. However, this is only part of the full story, because the probability of missing elements of the true active set is expected to increase with p.

Of course, it is possible to partition the data into $K > 2$ groups, say, each of size n/K, and estimate $\mathcal{A}$ by $\widehat{\mathcal{A}}^{(1)} \cap \widehat{\mathcal{A}}^{(2)} \cap \ldots \cap \widehat{\mathcal{A}}^{(K)}$, where $\widehat{\mathcal{A}}^{(k)}$ represents the estimated set of active indices from the kth partition. Such a variable selection procedure would be even more aggressive than the $K = 2$ version; improved bounds in Theorem 1 could be obtained, but the probability of missing true active indices would be increased. As the $K = 2$ procedure is already quite aggressive, we consider this to be the most natural choice in practice.

In the iterated version of this first variant of SIS, we apply SIS to each partition separately to obtain two sets of indices $\widehat{\mathcal{A}}_1^{(1)}$ and $\widehat{\mathcal{A}}_1^{(2)}$, each having k_1 elements. After forming the intersection $\widehat{\mathcal{A}}_1 = \widehat{\mathcal{A}}_1^{(1)} \cap \widehat{\mathcal{A}}_1^{(2)}$, we carry out penalized likelihood estimation as before to give a first approximation $\widehat{\mathcal{M}}_1$ to the true active set of features. We then perform a second stage of the ISIS procedure, as outlined in Section 2, to each partition separately to obtain sets of indices $\widehat{\mathcal{M}}_1 \cup \widehat{\mathcal{A}}_2^{(1)}$ and $\widehat{\mathcal{M}}_1 \cup \widehat{\mathcal{A}}_2^{(2)}$. Taking the intersection of these sets and re-estimating parameters using penalized likelihood as in Section 2 gives a second approximation $\widehat{\mathcal{M}}_2$ to the true active set. This process can be continued until we reach an iteration ℓ with $\widehat{\mathcal{M}}_\ell = \widehat{\mathcal{M}}_{\ell-1}$, or we have recruited d indices.

3.2 Second Variant of ISIS

Our second variant of SIS is a more conservative feature selection procedure and also relies on random partitioning the data into $K = 2$ groups as before. Again, we apply SIS to each partition separately, but now we recruit as many features into equal-sized sets of active indices $\widetilde{\mathcal{A}}^{(1)}$ and $\widetilde{\mathcal{A}}^{(2)}$ as are required to ensure that the intersection $\widetilde{\mathcal{A}} = \widetilde{\mathcal{A}}^{(1)} \cap \widetilde{\mathcal{A}}^{(2)}$ has d elements. We then apply a penalized pseudo-likelihood method to the features $X_{\tilde{\mathcal{A}}} = \{X_j : j \in \widetilde{\mathcal{A}}\}$ for final feature selection and parameter estimation.

Theoretical support for this method can be provided in the case of the linear model; namely, under certain regularity conditions, this variant of SIS possesses the sure screening property. More precisely, if Conditions (1)–(4) of Fan & Lv (2008) hold with $2\kappa + \tau < 1$, and we choose $d = \lfloor n/\log n \rfloor$, then there exists $C > 0$ such that

$$P(\mathcal{A} \subseteq \tilde{\mathcal{A}}) = 1 - O\{\exp(-Cn^{1-2\kappa}/\log n + \log p)\}.$$

The parameter $\kappa \geq 0$ controls the rate at which the minimum signal $\min_{j\in\mathcal{A}} |\beta_j|$ is allowed to converge to zero, while $\tau \geq 0$ controls the rate at which the maximal eigenvalue of the covariance matrix $\Sigma = \mathrm{Cov}(X_1, \ldots, X_p)$ is allowed to diverge to infinity. In fact, we insist that $\min_{j\in\mathcal{A}} |\beta_j| \geq n^{-\kappa}$ and $\lambda_{\max}(\Sigma) \leq n^{\tau}$ for large n, where $\lambda_{\max}(\Sigma)$ denotes the maximal eigenvalue of Σ. Thus, these technical conditions ensure that any non-zero signal is not too small, and that the features are not too close to being collinear, and the dimensionality is also controlled via $\log p = o(n^{1-2\kappa}/\log n)$, which is still of an exponential order. See Fan & Lv (2008) for further discussion of the sure screening property.

Recently, Fan & Song (2009) extended the result of Fan & Lv (2008) to generalized linear models. Let $\hat{L}_0 = \min_{\beta_0} n^{-1} \sum_{i=1}^n L(Y_i, \beta_0)$ be the baseline value to (2.1). The feature ranking procedure is equivalent to the thresholding method: $\widehat{\mathcal{M}}_{\nu_n} = \{j : L_j - L_0 \geq \nu_n\}$, in which ν_n is a given thresholding value. Under certainly regularity conditions, if

$$\min_{j\in\mathcal{A}} |\mathrm{cov}(X_j, Y)| \geq c_1 n^{-\kappa}, \qquad \text{for some } c_1 > 0 \text{ and } \kappa < 1/2$$

and $\nu_n = c_0 n^{-2\kappa}$ for a sufficiently small c_0, then we have

$$P(\mathcal{A} \subset \widehat{\mathcal{M}}_{\nu_n}) \to 1,$$

exponentially fast, provided that $\log p_n = o(n^{1-2\kappa})$. The sure screening property does not depend on the correlation of the features, as expected. However, the selected model size does depend on the correlation structure: The more correlated the features, the larger the selected model size. In fact, Fan & Song (2009) demonstrated further that with probability tending to one exponentially fast, $|\widehat{\mathcal{M}}_{\nu_n}| = O(\nu_n^{-2}\lambda_{\max}(\Sigma))$. When $\lambda_{\max}(\Sigma) = O(n^{\tau})$ and $\lambda_{\max}(\Sigma) = O(n^{-2\kappa})$, the selected model size is $|\widehat{\mathcal{M}}_{\nu_n}| = O(n^{2\kappa+\tau})$. In particularly, if the condition $2\kappa+\tau < 1$ is imposed as in Fan & Lv (2008), we can reduce safely the model size to $o(n)$ by independence learning.

An iterated version of this second variant of SIS is also available. At the first stage we apply SIS, taking enough features in equal-sized sets of active indices $\widetilde{\mathcal{A}}_1^{(1)}$ and $\widetilde{\mathcal{A}}_1^{(2)}$ to ensure that the intersection $\widetilde{\mathcal{A}}_1 = \widetilde{\mathcal{A}}_1^{(1)} \cap \widetilde{\mathcal{A}}_1^{(2)}$ has k_1 elements. Applying penalized likelihood to the features with indices in $\widetilde{\mathcal{A}}_1$ gives a first approximation $\widetilde{\mathcal{M}}_1$ to the true set of active indices. We then carry out a second stage of the ISIS procedure of Section 2 to each partition separately to obtain equal-sized new sets of indices $\widetilde{\mathcal{A}}_2^{(1)}$ and $\widetilde{\mathcal{A}}_2^{(2)}$, taking enough features to ensure that $\widetilde{\mathcal{A}}_2 = \widetilde{\mathcal{A}}_2^{(1)} \cap \widetilde{\mathcal{A}}_2^{(2)}$ has k_2 elements. Penalized likelihood applied to $\widetilde{\mathcal{M}}_1 \cap \widetilde{\mathcal{A}}_2$ gives a second approximation $\widetilde{\mathcal{M}}_2$ to the true set of active indices. As with the first variant, we continue until we reach an iteration ℓ with $\widetilde{\mathcal{M}}_\ell = \widetilde{\mathcal{M}}_{\ell-1}$, or we have recruited d indices.

4. Numerical Results

We illustrate the breadth of applicability of (I)SIS and its variants by studying its performance on simulated data in four different contexts: logistic regression, Poisson regression, robust regression (with a least absolute deviation criterion) and multi-class classification with support vector machines. We will consider three different configurations of the $p = 1000$ features $X_1, \ldots, X_p$:

Case 1: $X_1, \ldots, X_p$ are independent and identically distributed $N(0,1)$ random variables

Case 2: $X_1, \ldots, X_p$ are jointly Gaussian, marginally $N(0,1)$, and with $\mathrm{corr}(X_i, X_4) = 1/\sqrt{2}$ for all $i \neq 4$ and $\mathrm{corr}(X_i, X_j) = 1/2$ if i and j are distinct elements of $\{1, \ldots, p\} \setminus \{4\}$

Case 3: $X_1, \ldots, X_p$ are jointly Gaussian, marginally $N(0,1)$, and with $\mathrm{corr}(X_i, X_5) = 0$ for all $i \neq 5$, $\mathrm{corr}(X_i, X_4) = 1/\sqrt{2}$ for all $i \notin \{4,5\}$, and $\mathrm{corr}(X_i, X_j) = 1/2$ if i and j are distinct elements of $\{1, \ldots, p\} \setminus \{4,5\}$.

Case 1, with independent features, is the most straightforward for variable selection. In Cases 2 and 3, however, we have serial correlation such that $\mathrm{corr}(X_i, X_j)$ does not decay as $|i-j|$ increases. We will see later that for both Case 2 and Case 3 the true coefficients are chosen such that the response is marginally uncorrelated with X_4. We therefore expect feature selection in these situations to be more challenging, especially for the non-iterated versions of SIS. Notice that in the asymptotic theory of SIS in Fan & Lv (2008), this type of dependence is ruled out by their Condition (4).

Regarding the choice of d, the asymptotic theory of Fan & Lv (2008) shows that in the linear model there exists $\theta^* > 0$ such that we may obtain the sure screening property with $\lfloor n^{1-\theta^*} \rfloor <$

$d < n$. However, θ^* is unknown in practice, and therefore Fan and Lv recommend $d = \lfloor n/\log n \rfloor$ as a sensible choice. Of course, choosing a larger value of d increases the probability that SIS will include all of the correct variables, but including more inactive variables will tend to have a slight detrimental effect on the performance of the final variable selection and parameter estimation method. We have found that this latter effect is most noticeable in models where the response provides less information. In particular, the binary response of a logistic regression model and, to a lesser extent, the integer-valued response in a Poisson regression model are less informative than the real-valued response in a linear model. We therefore used $d = \lfloor \frac{n}{4\log n} \rfloor$ in the logistic regression and multicategory classification settings of Sections 4.1 and 4.5, $d = \lfloor \frac{n}{2\log n} \rfloor$ in the Poisson regression settings of Section 4.2 and $d = \lfloor \frac{n}{2} \rfloor$ in Section 4.4. These model-based, rather than data-adaptive, choices of d seem to be satisfactory, as the performance of the procedures is quite robust to different choices of d (in fact using $d = \lfloor \frac{n}{\log n} \rfloor$ for all models would still give good performance).

4.1 Logistic Regression

In this example, the data $(\mathbf{x}_1^T, Y_1), \ldots, (\mathbf{x}_n^T, Y_n)$ are independent copies of a pair $(\mathbf{x}^T, Y)$, where Y is distributed, conditional on $\mathbf{X} = \mathbf{x}$, as $\text{Bin}(1, p(\mathbf{x}))$, with $\log\big(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\big) = \beta_0 + \mathbf{x}^T\beta$. We choose $n = 400$.

As explained above, we chose $d = \lfloor \frac{n}{4\log n} \rfloor = 16$ in both the vanilla version of SIS outlined in Section 2 (Van-SIS), and the second variant (Var2-SIS) in Section 3.2. For the first variant (Var1-SIS), however, we used $d = \lfloor \frac{n}{\log n} \rfloor = 66$; note that since this means the selected features are in the intersection of two sets of size d, we typically end up with far fewer than d features selected by this method.

For the logistic regression example, the choice of final regularization parameter λ for the SCAD penalty (after all (I)SIS steps) was made by means of an independent validation data set of size n (generated from the same model as the original data, used only for tuning the parameters), rather than by cross-validation. This also applies for the LASSO and Nearest Shrunken Centroid (NSC, Tibshirani et al., 2003) methods which we include for comparison; instead of using SIS, this method regularizes the log-likelihood with an L_1-penalty. The reason for using the independent tuning data set is that the lack of information in the binary response means that cross-validation is particularly prone to overfitting in logistic regression, and therefore perfoms poorly for all methods.

The coefficients used in each of the three cases were as follows:

Case 1: $\beta_0 = 0, \beta_1 = 1.2439, \beta_2 = -1.3416, \beta_3 = -1.3500, \beta_4 = -1.7971, \beta_5 = -1.5810, \beta_6 = -1.5967$, and $\beta_j = 0$ for $j > 6$. The corresponding Bayes test error is 0.1368.

Case 2: $\beta_0 = 0, \beta_1 = 4, \beta_2 = 4, \beta_3 = 4, \beta_4 = -6\sqrt{2}$, and $\beta_j = 0$ for $j > 4$. The Bayes test error is 0.1074.

Case 3: $\beta_0 = 0, \beta_1 = 4, \beta_2 = 4, \beta_3 = 4, \beta_4 = -6\sqrt{2}, \beta_5 = 4/3$, and $\beta_j = 0$ for $j > 5$. The Bayes test error is 0.1040.

In Case 1, the coefficients were chosen randomly, and were generated as $(4\log n/\sqrt{n} + |Z|/4)U$ with $Z \sim N(0,1)$ and $U = 1$ with probability 0.5 and -1 with probability -0.5, independent of Z. For Cases 2 and 3, the choices ensure that even though $\beta_4 \neq 0$, we have that X_4 and Y are independent. The fact that X_4 is marginally independent of the response is designed to make it difficult for a popular method such as the two-sample t test or other independent learning methods

to recognize this important feature. Furthermore, for Case 3, we add another important variable X_5 with a small coefficient to make it even more difficult to identify the true model. For Case 2, the ideal variables picked up by the two sample test or independence screening technique are X_1, X_2 and X_3. Using these variables to build the ideal classifier, the Bayes risk is 0.3443, which is much larger than the Bayes error 0.1074 of the true model with X_1, X_2, X_3, X_4. In fact one may exaggerate Case 2 to make the Bayes error using the independence screening technique close to 0.5, which corresponds to random guessing, by setting $\beta_0 = 0$, $\beta_1 = \beta_2 = \beta_3 = a$, $\beta_m = a$ for $m = 5, 6, \cdots, j$, $\beta_4 = -a(j-1)\sqrt{2}/2$, and $\beta_m = 0$ for $m > j$. For example, the Bayes error using the independence screening technique, which deletes X_4, is 0.4290 when $j = 20$ and $a = 4$ while the corresponding Bayes error using X_m, $m = 1, 2, \cdots, 20$ is 0.0445.

In the tables below, we report several performance measures, all of which are based on 100 Monte Carlo repetitions. The first two rows give the median L_1 and squared L_2 estimation errors $\|\beta - \widehat{\beta}\|_1 = \sum_{j=0}^{p} |\beta_j - \hat{\beta}_j|$ and $\|\beta - \widehat{\beta}\|_2^2 = \sum_{j=0}^{p} (\beta_j - \hat{\beta}_j)^2$. The third row gives the proportion of times that the (I)SIS procedure under consideration includes all of the important features in the model, while the fourth reports the corresponding proportion of times that the final features selected, after application of the SCAD or LASSO penalty as appropriate, include all of the important ones. The fifth row gives the median final number of features selected. Measures of fit to the training data are provided in the sixth, seventh and eighth rows, namely the median values of $2Q(\hat{\beta}_0, \widehat{\beta})$, defined in (2.1), Akaike's information criterion (Akaike, 1974), which adds twice the number of features in the final model, and the Bayesian information criterion (Schwarz, 1978), which adds the product of $\log n$ and the number of features in the final model. Finally, an independent test data set of size $100n$ was used to evaluate the median value of $2Q(\hat{\beta}_0, \widehat{\beta})$ on the test data (Row 9), as well as to report the median 0-1 test error (Row 10), where we observe an error if the test response differs from the fitted response by more than $1/2$.

Table 1 compares five methods, Van-SIS, Var1-SIS, Var2-SIS, LASSO, and NSC. The most noticeable observation is that while the LASSO always includes all of the important features, it does so by selecting very large models—a median of 94 variables, as opposed to the correct number, 6, which is the median model size reported by all three SIS-based methods. This is due to the bias of the LASSO, as pointed out by Fan & Li (2001) and Zou (2006), which encourages the choice of a small regularization parameter to make the overall mean squared error small. Consequently, many unwanted features are also recruited. This is also evidenced by comparing the differences between L_1 and L_2 losses in the first two rows. Thus the LASSO method has large estimation error, and while $2Q(\hat{\beta}_0, \widehat{\beta})$ is small on the training data set, this is a result of overfit, as seen by the large values of AIC/BIC, $2Q(\hat{\beta}_0, \widehat{\beta})$ on the test data and the 0-1 test error.

As the features are independent in Case 1, it is unsurprising to see that Van-SIS has the best performance of the three SIS-based methods. Even with the larger value of d used for Var1-SIS, it tends to miss important features more often than the other methods. Although the method appears to have value as a means of obtaining a minimal set of features that should be included in a final model, we will not consider Var1-SIS further in our simulation study.

Table 2 displays the results of repeating the Case 1 simulations for Van-SIS, Var1-SIS and Var2-SIS under the same conditions, but using the LASSO penalty function rather than the SCAD penalty function after the SIS step. These versions are called Van-SIS-LASSO, Var1-SIS-LASSO and Var2-SIS-LASSO respectively. We see that, as well as decreasing the computational cost, using any of the three versions of SIS before the LASSO improves performance substantially compared with

applying the LASSO to the full set of features. On the other hand, the results are less successful than applying SIS and its variants in conjuction with the SCAD penalty for final feature selection and parameter estimation. We therefore do not consider Van-SIS-LASSO, Var1-SIS-LASSO and Var2-SIS-LASSO further.

	Van-SIS	Var1-SIS	Var2-SIS	LASSO	NSC
$\|\beta - \widehat{\beta}\|_1$	1.1093	1.2495	1.2134	8.4821	N/A
$\|\beta - \widehat{\beta}\|_2^2$	0.4861	0.5237	0.5204	1.7029	N/A
Prop. incl. (I)SIS models	0.99	0.84	0.91	N/A	N/A
Prop. incl. final models	0.99	0.84	0.91	1.00	0.34
Median final model size	6	6	6	94	3
$2Q(\hat{\beta}_0, \widehat{\beta})$ (training)	237.21	247.00	242.85	163.64	N/A
AIC	250.43	259.87	256.26	352.54	N/A
BIC	277.77	284.90	282.04	724.70	N/A
$2Q(\hat{\beta}_0, \widehat{\beta})$ (test)	271.81	273.08	272.91	318.52	N/A
0-1 test error	0.1421	0.1425	0.1426	0.1720	0.3595

Table 1: Logistic regression, Case 1

	Van-SIS-LASSO	Var1-SIS-LASSO	Var2-SIS-LASSO
$\|\beta - \widehat{\beta}\|_1$	3.8500	2.1050	3.0055
$\|\beta - \widehat{\beta}\|_2^2$	1.0762	0.7536	0.9227
Prop. incl. (I)SIS models	0.99	0.84	0.91
Prop. incl. final models	0.99	0.84	0.91
Median final model size	16.0	9.0	14.5
$2Q(\hat{\beta}_0, \widehat{\beta})$ (training)	207.86	240.44	226.95
AIC	239.69	260.49	255.99
BIC	302.98	295.40	316.36
$2Q(\hat{\beta}_0, \widehat{\beta})$ (test)	304.79	280.95	291.79
0-1 test error	0.1621	0.1476	0.1552

Table 2: Logistic regression, Case 1

In Cases 2 and 3, we also consider the iterated versions of Van-SIS and Var2-SIS, which we denote Van-ISIS and Var2-ISIS respectively. At each intermediate stage of the ISIS procedures, the Bayesian information criterion was used as a fast way of choosing the SCAD regularization parameter.

From Tables 3 and 4, we see that the non-iterated SIS methods fail badly in these awkward cases. Their performance is similar to that of the LASSO method. On the other hand, both of the iterated methods Van-ISIS and Var2-ISIS perform extremely well (and similarly to each other).

4.2 Poisson Regression

In our second example, the generic response Y is distributed, conditional on $\mathbf{X} = \mathbf{x}$, as Poisson$(\mu(\mathbf{x}))$, where $\log \mu(\mathbf{x}) = \beta_0 + \mathbf{x}^T\beta$.

	Van-SIS	Van-ISIS	Var2-SIS	Var2-ISIS	LASSO	NSC
$\|\beta - \widehat{\beta}\|_1$	20.0504	1.9445	20.1100	1.8450	21.6437	N/A
$\|\beta - \widehat{\beta}\|_2^2$	9.4101	1.0523	9.3347	0.9801	9.1123	N/A
Prop. incl. (I)SIS models	0.00	1.00	0.00	1.00	N/A	N/A
Prop. incl. final models	0.00	1.00	0.00	1.00	0.00	0.21
Median final model size	16	4	16	4	91	16.5
$2Q(\hat{\beta}_0, \widehat{\beta})$ (training)	307.15	187.58	309.63	187.42	127.05	N/A
AIC	333.79	195.58	340.77	195.58	311.10	N/A
BIC	386.07	211.92	402.79	211.55	672.34	N/A
$2Q(\hat{\beta}_0, \widehat{\beta})$ (test)	344.25	204.23	335.21	204.28	258.65	N/A
0-1 test error	0.1925	0.1092	0.1899	0.1092	0.1409	0.3765

Table 3: Logistic regression, Case 2

	Van-SIS	Van-ISIS	Var2-SIS	Var2-ISIS	LASSO	NSC
$\|\beta - \widehat{\beta}\|_1$	20.5774	2.6938	20.6967	3.2461	23.1661	N/A
$\|\beta - \widehat{\beta}\|_2^2$	9.4568	1.3615	9.3821	1.5852	9.1057	N/A
Prop. incl. (I)SIS models	0.00	1.00	0.00	1.00	N/A	N/A
Prop. incl. final models	0.00	0.90	0.00	0.98	0.00	0.17
Median final model size	16	5	16	5	101.5	10
$2Q(\hat{\beta}_0, \widehat{\beta})$ (training)	269.20	187.89	296.18	187.89	109.32	N/A
AIC	289.20	197.59	327.66	198.65	310.68	N/A
BIC	337.05	218.10	389.17	219.18	713.78	N/A
$2Q(\hat{\beta}_0, \widehat{\beta})$ (test)	360.89	225.15	358.13	226.25	275.55	N/A
0-1 test error	0.1933	0.1120	0.1946	0.1119	0.1461	0.3866

Table 4: Logistic regression, Case 3

Due to the extra information in the count response, we choose $n = 200$, and apply all versions of (I)SIS with $d = \lfloor \frac{n}{2 \log n} \rfloor = 37$. We also use 10-fold cross-validation to choose the final regularization parameter for the SCAD and LASSO penalties. The coefficients used were as follows:

Case 1: $\beta_0 = 5, \beta_1 = -0.5423, \beta_2 = 0.5314, \beta_3 = -0.5012, \beta_4 = -0.4850, \beta_5 = -0.4133, \beta_6 = 0.5234$, and $\beta_j = 0$ for $j > 6$.

Case 2: $\beta_0 = 5, \beta_1 = 0.6, \beta_2 = 0.6, \beta_3 = 0.6, \beta_4 = -0.9\sqrt{2}$, and $\beta_j = 0$ for $j > 4$.

Case 3: $\beta_0 = 5, \beta_1 = 0.6, \beta_2 = 0.6, \beta_3 = 0.6, \beta_4 = -0.9\sqrt{2}, \beta_5 = 0.15$, and $\beta_j = 0$ for $j > 5$.

In Case 1, the magnitudes of the coefficients $\beta_1, \ldots, \beta_6$ were generated as $(\frac{\log n}{\sqrt{n}} + |Z|/8)U$ with $Z \sim N(0,1)$ and $U = 1$ with probability 0.5 and -1 with probability 0.5, independently of Z. Again, the choices in Cases 2 and 3 ensure that, even though $\beta_4 \neq 0$, we have $\mathrm{corr}(X_4, Y) = 0$. The coefficients are a re-scaled version of those in the logistic regression model, except that $\beta_0 = 5$ is used to control an appropriate signal-to-noise ratio.

The results are shown in Tables 5, 6 and 7. Even in Case 1, with independent features, the ISIS methods outperform SIS, so we chose not to present the results for SIS in the other two cases. Again,

both Van-ISIS and Var2-ISIS perform extremely well, almost always including all the important features in relatively small final models. The LASSO method continues to suffer from overfitting, particularly in the difficult Cases 2 and 3.

	Van-SIS	Van-ISIS	Var2-SIS	Var2-ISIS	LASSO
$\|\beta - \widehat{\beta}\|_1$	0.0695	0.1239	1.1773	0.1222	0.1969
$\|\beta - \widehat{\beta}\|_2^2$	0.0225	0.0320	0.4775	0.0330	0.0537
Prop. incl. (I)SIS models	0.76	1.00	0.45	1.00	N/A
Prop. incl. final models	0.76	1.00	0.45	1.00	1.00
Median final model size	12	18	13	17	27
$2Q(\hat{\beta}_0, \widehat{\beta})$ (training)	1560.85	1501.80	7735.51	1510.38	1534.19
AIC	1586.32	1537.80	7764.51	1542.14	1587.23
BIC	1627.06	1597.17	7812.34	1595.30	1674.49
$2Q(\hat{\beta}_0, \widehat{\beta})$ (test)	1557.74	1594.10	14340.26	1589.51	1644.63

Table 5: Poisson regression, Case 1

	Van-ISIS	Var2-ISIS	LASSO
$\|\beta - \widehat{\beta}\|_1$	0.2705	0.2252	3.0710
$\|\beta - \widehat{\beta}\|_2^2$	0.0719	0.0667	1.2856
Prop. incl. (I)SIS models	1.00	0.97	N/A
Prop. incl. final models	1.00	0.97	0.00
Median final model size	18	16	174
$2Q(\hat{\beta}_0, \widehat{\beta})$ (training)	1494.53	1509.40	1369.96
AIC	1530.53	1541.17	1717.91
BIC	1589.90	1595.74	2293.29
$2Q(\hat{\beta}_0, \widehat{\beta})$ (test)	1629.49	1614.57	2213.10

Table 6: Poisson regression, Case 2

	Van-ISIS	Var2-ISIS	LASSO
$\|\beta - \widehat{\beta}\|_1$	0.2541	0.2319	3.0942
$\|\beta - \widehat{\beta}\|_2^2$	0.0682	0.0697	1.2856
Prop. incl. (I)SIS models	0.97	0.91	0.00
Prop. incl. final models	0.97	0.91	0.00
Median final model size	18	16	174
$2Q(\hat{\beta}_0, \widehat{\beta})$ (training)	1500.03	1516.14	1366.63
AIC	1536.03	1546.79	1715.35
BIC	1595.40	1600.17	2293.60
$2Q(\hat{\beta}_0, \widehat{\beta})$ (test)	1640.27	1630.58	2389.09

Table 7: Poisson regression, Case 3

4.3 Robust Regression

We have also conducted similar numerical experiments using L_1-regression for the three cases in an analogous manner to the previous two examples. We obtain similar results. Both versions of ISIS are effective in selecting important features with relatively low false positive rates. Hence, the prediction errors are also small. On the other hand, LASSO missed the difficult variables in cases 2 and 3 and also selected models with a large number of features to attenuate the bias of the variable selection procedure. As a result, its prediction errors are much larger. To save space, we omit the details of the results.

4.4 Linear Regression

Note that our new ISIS procedure allows feature deletion in each step. It is an important improvement over the original proposal of Fan & Lv (2008) even in the ordinary least-squares setting. To demonstrate this, we choose Case 3, the most difficult one, with coefficients given as follows.

Case 3: $\beta_0 = 0, \beta_1 = 5, \beta_2 = 5, \beta_3 = 5, \beta_4 = -15\sqrt{2}/2, \beta_5 = 1$, and $\beta_j = 0$ for $j > 5$.

The response Y is set as $Y = \mathbf{x}^T\beta + \varepsilon$ with independent $\varepsilon \sim N(0,1)$. This model is the same as Example 4.2.3 of Fan & Lv (2008). Using $n = 70$ and $d = n/2$, our new ISIS method includes all five important variables for 91 out of the 100 repetitions, while the original ISIS without feature deletion includes all the important features for only 36 out of the 100 repetitions. The median model size of our new variable selection procedure with variable deletion is 21, whereas the median model size corresponding to the original ISIS of Fan & Lv (2008) is 19.

We have also conducted the numerical experiment with a different sample size $n = 100$ and $d = n/2 = 50$. For 97 out of 100 repetitions, our new ISIS includes all the important features while ISIS without variable deletion includes all the important features for only 72 repetitions. Their median model sizes are both 26. This clearly demonstrates the improvement of allowing feature deletion in this example.

4.5 Multicategory Classification

Our final example in this section is a four-class classification problem. Here we study two different feature configurations, both of which depend on first generating independent $\tilde{X}_1, \ldots, \tilde{X}_p$ such that $\tilde{X}_1, \ldots, \tilde{X}_4$ are uniformly distributed on $[-\sqrt{3}, \sqrt{3}]$, and $\tilde{X}_5, \ldots, \tilde{X}_p$ are distributed as $N(0,1)$. We use these random variables to generate the following cases:

Case 1: $X_j = \tilde{X}_j$ for $j = 1, \ldots, p$

Case 2: $X_1 = \tilde{X}_1 - \sqrt{2}\tilde{X}_5, X_2 = \tilde{X}_2 + \sqrt{2}\tilde{X}_5, X_3 = \tilde{X}_3 - \sqrt{2}\tilde{X}_5, X_4 = \tilde{X}_4 + \sqrt{2}\tilde{X}_5$, and $X_j = \sqrt{3}\tilde{X}_j$ for $j = 5, \ldots, p$.

Conditional on $\mathbf{X} = \mathbf{x}$, the response Y was generated according to $P(Y = k|\widetilde{\mathbf{X}} = \tilde{\mathbf{x}}) \propto \exp\{f_k(\tilde{\mathbf{x}})\}$, for $k = 1, \ldots, 4$, where $f_1(\tilde{\mathbf{x}}) = -a\tilde{x}_1 + a\tilde{x}_4$, $f_2(\tilde{\mathbf{x}}) = a\tilde{x}_1 - a\tilde{x}_2$, $f_3(\tilde{\mathbf{x}}) = a\tilde{x}_2 - a\tilde{x}_3$ and $f_4(\tilde{\mathbf{x}}) = a\tilde{x}_3 - a\tilde{x}_4$ with $a = 5/\sqrt{3}$.

In both Case 1 and Case 2, all features have the same standard deviation since $\mathrm{sd}(X_j) = 1$ for $j = 1, 2, \cdots, p$ in Case 1 and $\mathrm{sd}(X_j) = \sqrt{3}$ for $j = 1, 2, \cdots, p$ in Case 2. Moreover, for this case, the variable X_5 is marginally unimportant, but jointly significant, so it represents a challenge to identify this as an important variable. For both Case 1 and Case 2, the Bayes error is 0.1373.

For the multicategory classification we use the loss function proposed by Lee, Lin and Wahba (2004). Denote the coefficients for the kth class by β_{0k} and β_k for $k = 1,2,3,4$, and let $\mathbf{B} = ((\beta_{01}, \beta_1^T)^T,$
$(\beta_{02}, \beta_2^T)^T, (\beta_{03}, \beta_3^T)^T, (\beta_{04}, \beta_4^T)^T)$. Let $f_k(\mathbf{x}) \equiv f_k(\mathbf{x}, \beta_{0k}, \beta_k) = \beta_{0k} + \mathbf{x}^T\beta_k$, $k = 1,2,3,4$, and

$$\mathbf{f}(\mathbf{x}) \equiv \mathbf{f}(\mathbf{x}, \mathbf{B}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}), f_4(\mathbf{x}))^T.$$

The loss function is given by $L(Y, \mathbf{f}(\mathbf{x})) = \sum_{j \neq Y} [1 + f_j(\mathbf{x})]_+$, where $[\psi]_+ = \psi$ if $\psi \geq 0$ and 0 otherwise. Deviating slightly from our standard procedure, the marginal utility of the j^{th} feature is defined by

$$L_j = \min_{\mathbf{B}} \sum_{i=1}^{n} L(Y_i, \mathbf{f}(X_{ij}, \mathbf{B})) + \frac{1}{2} \sum_{k=1}^{4} \beta_{jk}^2$$

to avoid possible unidentifiablity issues due to the hinge loss function. Analogous modification is applied to (3) in the iterative feature selection step. With estimated coefficients $\hat{\beta}_{0k}$ and $\widehat{\beta}_k$, and $\hat{f}_k(\mathbf{x}) = \hat{\beta}_{0k} + \mathbf{x}^T\widehat{\beta}_k$ for $k = 1,2,3,4$, the estimated classification rule is given by $\text{argmax}_k \hat{f}_k(\mathbf{x})$. There are some other appropriate multi-category loss functions such as the one proposed by Liu, Shen and Doss (2005).

As with the logistic regression example in Section 4.1, we use $n = 400$, $d = \lfloor \frac{n}{4 \log n} \rfloor = 16$ and an independent validation data set of size n to pick the final regularization parameter for the SCAD penalty.

The results are given in Table 8. The mean estimated testing error was based on a further testing data set of size $200n$, and we also report the standard error of this mean estimate. In the case of independent features, all (I)SIS methods have similar performance. The benefits of using iterated versions of the ISIS methodology are again clear for Case 2, with dependent features.

	Van-SIS	Van-ISIS	Var2-SIS	Var2-ISIS	LASSO	NSC
	Case 1					
Prop. incl. (I)SIS models	1.00	1.00	0.99	1.00	N/A	N/A
Prop. incl. final model	1.00	1.00	0.99	1.00	0.00	0.68
Median modal size	2.5	4	10	5	19	4
0-1 test error	0.3060	0.3010	0.2968	0.2924	0.3296	0.4524
Test error standard error	0.0067	0.0063	0.0067	0.0061	0.0078	0.0214
	Case 2					
Prop. incl. (I)SIS models	0.10	1.00	0.03	1.00	N/A	N/A
Prop. incl. final models	0.10	1.00	0.03	1.00	0.33	0.30
Median modal size	4	11	5	9	54	9
0-1 test error	0.4362	0.3037	0.4801	0.2983	0.4296	0.6242
Test error standard error	0.0073	0.0065	0.0083	0.0063	0.0043	0.0084

Table 8: Multicategory classification

5. Real Data Examples

In this section, we apply our proposed methods to two real data sets. The first one has a binary response while the second is multi-category. We treat both as classification problems and use the

hinge loss discussed in Section 4.5. We compare our methods with two alternatives: the LASSO and NSC.

5.1 Neuroblastoma Data

We first consider the neuroblastoma data used in Oberthuer et al. (2006). The study consists of 251 patients of the German Neuroblastoma Trials NB90-NB2004, diagnosed between 1989 and 2004. At diagnosis, patients' ages range from 0 to 296 months with a median age of 15 months. They analyzed 251 neuroblastoma specimens using a customized oligonucleotide microarray with the goal of developing a gene expression-based classification rule for neuroblastoma patients to reliably predict courses of the disease. This also provides a comprehensive view on which set of genes is responsible for neuroblastoma.

The complete data set, obtained via the MicroArray Quality Control phase-II (MAQC-II) project, includes gene expression over 10,707 probe sites. Of particular interest is to predict the response labeled "3-year event-free survival" (3-year EFS) which is a binary variable indicating whether each patient survived 3 years after the diagnosis of neuroblastoma. Excluding five outlier arrays, there are 246 subjects out of which 239 subjects have 3-year EFS information available with 49 positives and 190 negatives. We apply SIS and ISIS to reduce dimensionality from $p = 10,707$ to $d = 50$. On the other hand, our competitive methods LASSO and NSC are applied directly to $p = 10,707$ genes. Whenever appropriate, five-fold cross validation is used to select tuning parameters. We randomly select 125 subjects (25 positives and 100 negatives) to be the training set and the remainder are used as the testing set. Results are reported in the top half of Table 9. Selected probes for LASSO and all different (I)SIS methods are reported in Table 10.

In MAQC-II, a specially designed end point is the gender of each subject, which should be an easy classification. The goal of this specially designed end point is to compare the performance of different classifiers for simple classification jobs. The gender information is available for all the non-outlier 246 arrays with 145 males and 101 females. We randomly select 70 males and 50 females to be in the training set and use the others as the testing set. We set $d = 50$ for our SIS and ISIS as in the case of the 3-year EFS end point. The results are given in the bottom half of Table 9. Selected probes for all different methods are reported in Table 11.

End point		SIS	ISIS	var2-SIS	var2-ISIS	LASSO	NSC
3-year EFS	No. of features	5	23	10	12	57	9413
	Testing error	19/114	22/114	22/114	21/114	22/114	24/114
Gender	No. of features	6	2	4	2	42	3
	Testing error	4/126	4/126	4/126	4/126	5/126	4/126

Table 9: Results from analyzing two endpoints of the neuroblastoma data

We can see from Table 9 that our (I)SIS methods compare favorably with the LASSO and NSC. Especially for the end point 3-year EFS, our methods use fewer features while giving smaller testing error. For the end point GENDER, Table 11 indicates that the most parsimonious model given by ISIS and Var2-ISIS is a sub model of others.

5.2 SRBCT Data

In this section, we apply our method to the children cancer data set reported in Khan et al. (2001). Khan et al. (2001) used artificial neural networks to develop a method of classifying the small, round

Probe	SIS	ISIS	var2-SIS	var2-ISIS	LASSO
'A_23_P160638'					x
'A_23_P168916'		x			x
'A_23_P42882'		x			
'A_23_P145669'					x
'A_32_P50522'					x
'A_23_P34800'					x
'A_23_P86774'		x			
'A_23_P417918'			x		x
'A_23_P100711'					x
'A_23_P145569'					x
'A_23_P337201'					x
'A_23_P56630'		x		x	x
'A_23_P208030'		x			x
'A_23_P211738'		x			
'A_23_P153692'					x
'A_24_P148811'			x		
'A_23_P126844'			x		x
'A_23_P25194'					x
'A_24_P399174'					x
'A_24_P183664'					x
'A_23_P59051'				x	
'A_24_P14464'					x
'A_23_P501831'	x		x		
'A_23_P103631'				x	
'A_23_P32558'			x		
'A_23_P25873'		x			
'A_23_P95553'					x
'A_24_P227230'		x			x
'A_23_P5131'					x
'A_23_P218841'					x
'A_23_P58036'					x
'A_23_P89910'		x			
'A_24_P98783'					x
'A_23_P121987'		x			x
'A_32_P365452'					x
'A_23_P109682'		x			
'Hs58251.2'				x	
'A_23_P121102'		x			
'A_23_P3242'					x
'A_32_P177667'					x
'Hs6806.2'					x
'Hs376840.2'					x
'A_24_P136691'					x
'Pro25G_B35_D_7'		x		x	
'A_23_P87401'			x		
'A_32_P302472'					x
'Hs343026.1'				x	
'A_23_P216225'		x		x	x
'A_23_P203419'		x			x
'A_24_P22163'		x			x
'A_24_P187706'					x
'C1_QC'					x
'Hs190380.1'		x			x
'Hs117120.1'				x	
'A_32_P133518'					x
'EQCP1_Pro25G_T5'					x
'A_24_P111061'				x	
'A_23_P20823'	x	x		x	x
'A_24_P211151'			x		
'Hs265827.1'		x			x
'Pro25G_B12_D_7'					x
'Hs156406.1'					x
'A_24_P902509'				x	
'A_32_P32653'					x
'Hs42896.1'		x			
'A_32_P143793'	x		x		x
'A_23_P391382'					x
'A_23_P327134'					x
'Pro25G_EQCP1_T5'					x
'A_24_P351451'			x		
'Hs170298.1'					x
'A_23_P159390'					x
'Hs272191.1'		x			
'r60_a135'					x
'Hs439489.1'					x
'A_23_P107295'					x
'A_23_P100764'	x	x	x	x	x
'A_23_P157027'		x			
'A_24_P342055'					x
'A_23_P1387'	x				
'Hs6911.1'					x
'r60_1'					x

Table 10: Selected probes for the 3-year EFS end point

Probe	SIS	ISIS	var2-SIS	var2-ISIS	LASSO	NSC
'A_23_P201035'					x	
'A_24_P167642'					x	
'A_24_P55295'					x	
'A_24_P82200'	x					
'A_23_P109614'					x	
'A_24_P102053'					x	
'A_23_P170551'					x	
'A_23_P329835'						x
'A_23_P70571'					x	
'A_23_P259901'					x	
'A_24_P222000'					x	
'A_23_P160729'					x	
'A_23_P95553'	x		x			
'A_23_P100315'					x	
'A_23_P10172'					x	
'A_23_P137361'					x	
'A_23_P202484'					x	
'A_24_P56240'					x	
'A_32_P104448'					x	
'(-)3xSLv1'					x	
'A_24_P648880'					x	
'Hs446389.2'					x	
'A_23_P259314'	x	x	x	x	x	x
'Hs386420.1'					x	
'Pro25G_B32_D_7'					x	
'Hs116364.2'					x	
'A_32_P375286'	x				x	
'A_32_P152400'					x	
'A_32_P105073'					x	
'Hs147756.1'	x					
'Hs110039.1'					x	
'r60_a107'					x	
'Hs439208.1'					x	
'A_32_P506090'					x	
'A_24_P706312'			x			
'Hs58042.1'					x	
'A_23_P128706'					x	
'Hs3569.1'					x	
'A_24_P182900'					x	
'A_23_P92042'					x	
'Hs170499.1'					x	
'A_24_P500584'	x	x	x	x	x	x
'A_32_P843590'					x	
'Hs353080.1'					x	
'A_23_P388200'					x	
'C1_QC'					x	
'Hs452821.1'					x	

Table 11: Selected probe for Gender end point

blue cell tumors (SRBCTs) of childhood to one of the four categories: neuroblastoma (NB), rhabdomyosarcoma (RMS), non-Hodgkin lymphoma (NHL), and the Ewing family of tumors (EWS) using cDNA gene expression profiles. Accurate diagnosis of SRBCTs to these four distinct diagnostic categories is important in that the treatment options and responses to therapy are different from one category to another.

After filtering, 2308 gene profiles out of 6567 genes are given in the SRBCT data set. It is available online at http://research.nhgri.nih.gov/microarray/Supplement/. It includes a training set

of size 63 (12 NBs, 20 RMSs, 8 NHLs, and 23 EWS) and an independent test set of size 20 (6 NBs, 5 RMSs, 3 NHLs, and 6 EWS).

Before performing classification, we standardize the data sets by applying a simple linear transformation to both the training set and the test set. The linear transformation is based on the training data so that, after standardizing, the training data have mean zero and standard deviation one. Our (I)SIS reduces dimensionality from $p = 2308$ to $d = \lfloor 63/\log 63 \rfloor = 15$ first while alternative methods LASSO and NSC are applied to $p = 2308$ gene directly. Whenever appropriate, a four-fold cross validation is used to select tuning parameters.

ISIS, var2-ISIS, LASSO and NSC all achieve zero test error on the 20 samples in the test set. NSC uses 343 genes and LASSO requires 71 genes. However ISIS and var2-ISIS use 15 and 14 genes, respectively.

This real data application delivers the same message that our new ISIS and var2-ISIS methods can achieve competitive classification performance using fewer features.

Acknowledgments

Jianqing Fan's research was partially supported by NSF Grants DMS-0704337 and DMS-0714554 and NIH Grant R01-GM072611. Yichao Wu would like to acknowledge the support National Science Foundation grant DMS-0905561 and NCSU Faculty Research and Professional Development Award.

References

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

Hussein Almuallim and Thomas G. Diettrich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1–2):279–305, 1994.

Anestis Antoniadis and Jianqing Fan. Regularized wavelet approximations (with discussion). *J. Amer. Statist. Assoc.*, 96(455):939-967, 2001.

Eric Bair, Trevor Hastie, Debashis Paul and Robert Tibshirani. Prediction by supervised principal components. *J. Amer. Statist. Assoc.*, 101(473):119-137, 2006.

Yoshua Bengio and Nicolas Chapados. Extensions to metric based model selection. *J. Mach. Learn. Res.*, 3:1209–1227, 2003.

Jinbo Bi, Kristin P. Bennett, Mark J. Embrechts, Curt M. Breneman and Minghu Song. Dimensionality reduction via sparse support vector machines. *J. Mach. Learn. Res.* 3:1229–1243, 2003.

Peter J. Bickel, Ya'acov Ritov and Alexandre Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Ann. Statist.*, 37(4):1705–1732, 2009.

Emmanuel Candes and Terrence Tao. The Dantzig selector: statistical estimation when p is much larger than n (with discussion). *Ann. Statist.*, 35(6):2313-2404, 2007.

David L. Donoho and Michael Elad. Maximal sparsity representation via L_1 Minimization. *Proc. Nat. Aca. Sci.*, 100:2197–2202, 2003.

Sandrine Dudoit, Juliet P. Shaffer and Jennifer C. Boldrick. Multiple hypothesis testing in microarray experiments. *Statist. Sci.*, 18:71–103, 2003.

Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani. Least angle regression (with discussion). *Ann. Statist.*, 32: 409–499, 2004.

Bradley Efron. Microarrays, empirical Bayes and the two-groups model (with discussion). *Statist. Sci.*, 23:1–47, 2008.

Jianqing Fan and Yingying Fan. High dimensional classification using shrunken independence rule. *Ann. Statist*, 36(6):3605–2637, 2008.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Amer. Statist. Assoc.*, 96:1348–1360, 2001.

Jianqing Fan and Jinchi Lv. Sure independence screening for ultra-high dimensional feature space (with discussion). *J. Roy. Statist. Soc., Ser. B*, 70:849–911, 2008.

Jianqing Fan and Heng Peng. On non-concave penalized likelihood with diverging number of parameters. *Ann. Statist.*, 32(3):928–961, 2004.

Jianqing Fan and Yi Ren. Statistical analysis of DNA microarray data. *Clinical Cancer Res.*, 12:4469–4473, 2006.

Jianqing Fan and Rui Song. Sure Independence Screening in Generalized Linear Models with NP-Dimensionality. *Manuscript*, 2009.

Yoav Freund, and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. System Sci.*, 55(1):119–139.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

Isabelle Guyon, Steve Gunn, Masoud Nikravesh and Lofti Zadeh, editors. *Feature Extraction, Foundations and Applications*. Springer, New York, 2006.

Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *International Conference on Machine Learning*, Stanford, CA, pages 359–366, 2000.

Peter Hall, D. M. Titterington, and Jing-Hao Xue. Tiling methods for assessing the influence of components in a classifier. *J. Roy. Statist. Soc., Ser. B*, 71(4):783–803, 2009.

Trevor Hastie, Robert Tibshirani and Jerome Friedman. *The elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2001.

Peter J. Huber. Robust estimation of location. *Ann. Math. Statist.*, 35:73–101, 1964.

Javed Khan, Jun S. Wei, Markus Ringnér, Lao H. Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Manfred Schwab, Cristina R. Antonescu, Carsten Peterson and Paul S. Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7:673–679, 2001.

Igor Kononenko. Estimating attributes: Analysis and extension of RELIEF. In *Machine Learning: ECML-94*, Springer Berlin/Heidelberg, 1994.

Yoonkyung Lee, Yi Lin and Grace Wahba. Multicategory Support Vector Machines, Theory, and Application to the Classification of Microarray Data and Satellite Radiance Data. *J. Amer. Statist. Assoc.*, 99(465):67–81, 2004.

Huan Liu and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Boston MA, 1998.

Yufeng Liu, Xiaotong Shen and Hani Doss. Multicategory ψ-learning and support vector machine: computational tools. *J. Computat. Graph. Statist.*, 14(1):219–236, 2005.

Peter McCullagh and John A. Nelder. *Generalized Linear Models*. Chapman & Hall, London, 1989.

Nicolai Meinshausen and Peter Bühlmann. High dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006.

André Oberthuer, Frank Berthold, Patrick Warnat, Barbara Hero, Yvonne Kahlert, Rüdiger Spitz, Karen Ernestus, Rainer König, Stefan Haas, Roland Eils, Manfred Schwab, Benedikt Brors, Frank Westermann, Matthias Fischer. Customized Oligonucleotide Microarray Gene Expression Based Classification of Neuroblastoma Patients Outperforms Current Clinical Risk Stratification. *Journal of Clinical Oncology*, 24:5070–5078, 2006.

Mee Young Park and Trevor Hastie. L_1-regularization path algorithm for generalized linear models. *J. Roy. Statist. Soc. Ser. B*, 69(4):659–677, 2007.

Debashis Paul, Eric Bair, Trevor Hastie and Robert Tibshirani. "Pre-conditioning" for feature selection and regression in high-dimensional problems. *Ann. Statist.*, 36(4):1595–1618.

Gideon Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6(2):461–464, 1978.

Robert Tibshirani. Regression shrinkage and selection via lasso. *Jour. Roy. Statist. Soc. B.*, 58(1):267–288, 1996.

Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan and Gilbert Chu. Class prediction by nearest shrunken centroids, with applications to DNA microarrays. *Statist. Sci.*, 18(1):104–117, 2003.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

Lei Yu and Huan Liu. Feature selection for high-dimensional data: a fast correlation-based filter solution. In *International Conference on Machine Learning*, pages 856–863, Washington DC, USA, 2003.

Hui Zou. The adaptive Lasso and its oracle properties. *J. Amer. Statist. Assoc*., 101(476):1418–1429, 2006.

Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *Ann. Statist*., forthcoming.

Cun-Hui Zhang and Jian Huang. The sparsity and bias of the LASSO selection in high-dimensional linear regression. *Ann. Statist*., 36(4):1567–1594.

Zheng Zhao and Huan Liu. Searching for interacting features. In *Proceedings of the International joint conferences on Artificial Intelligence*, pages 1156–1161, Hyderabad, India, 2007.

Peng Zhao and Bin Yu. On model selection consistency of Lasso. *J. Machine Learning Res*., 7:2541–2563, 2006.

Hui Zou and Runze Li. One-step sparse estimates in nonconcave penalized likelihood models (with discussion). *Ann. Statist*., 36(4):1509-1566, 2008.

Journal of Machine Learning Research 10 (2009) 2039-2078 Submitted 12/08; Revised 3/09; Published 9/09

Evolutionary Model Type Selection for Global Surrogate Modeling

Dirk Gorissen DIRK.GORISSEN@UGENT.BE
Tom Dhaene TOM.DHAENE@UGENT.BE
Filip De Turck FILIP.DETURCK@UGENT.BE
Ghent University - IBBT
Department of Information Technology (INTEC)
Gaston Crommenlaan 8 bus 201
9050 Gent, Belgium

Editor: Melanie Mitchell

Abstract

Due to the scale and computational complexity of currently used simulation codes, global surrogate (metamodels) models have become indispensable tools for exploring and understanding the design space. Due to their compact formulation they are cheap to evaluate and thus readily facilitate visualization, design space exploration, rapid prototyping, and sensitivity analysis. They can also be used as accurate building blocks in design packages or larger simulation environments. Consequently, there is great interest in techniques that facilitate the construction of such approximation models while minimizing the computational cost and maximizing model accuracy. Many surrogate model types exist (Support Vector Machines, Kriging, Neural Networks, etc.) but no type is optimal in all circumstances. Nor is there any hard theory available that can help make this choice. In this paper we present an automatic approach to the model type selection problem. We describe an adaptive global surrogate modeling environment with adaptive sampling, driven by speciated evolution. Different model types are evolved cooperatively using a Genetic Algorithm (heterogeneous evolution) and compete to approximate the iteratively selected data. In this way the optimal model type and complexity for a given data set or simulation code can be dynamically determined. Its utility and performance is demonstrated on a number of problems where it outperforms traditional sequential execution of each model type.

Keywords: model type selection, genetic algorithms, global surrogate modeling, function approximation, active learning, adaptive sampling

1. Introduction

For many problems from science and engineering it is impractical to perform experiments on the physical world directly (e.g., airfoil design, earthquake propagation). Instead, complex, physics-based simulation codes are used to run experiments on computer hardware. While allowing scientists more flexibility to study phenomena under controlled conditions, computer experiments require a substantial investment of computation time. One simulation may take many minutes, hours, days or even weeks. A simpler approximation of the simulator is needed to make sensitivity analysis, visualization, design space exploration, etc. feasible (Forrester et al., 2008; Simpson et al., 2008).

As a result researchers have turned to various approximation methods that mimic the behavior of the simulation model as closely as possible while being computationally cheap(er) to evaluate. Different types of approximation methods exist, each with their relative strengths. This work con-

centrates on the use of data-driven, global approximations using compact surrogate models (also known as emulators, metamodels or response surface models) in the context of computer experiments. The objective is to construct a high fidelity approximation model that is as accurate as possible over the *complete* design space of interest using as little simulation points as possible. Once constructed, the global surrogate model (also referred to as a replacement metamodel[1]) is reused in other stages of the computational science and engineering pipeline. So optimization is not the main goal, but rather a useful post processing step.

The primary users of global surrogate modeling methods are domain experts, few of which will be experts in the intricacies of efficient sampling and modeling strategies. Their primary concern is obtaining an accurate replacement metamodel for their problem as fast as possible and with minimal overhead. Model (type) selection, model parameter optimization, sampling strategy, etc. are of lesser or no interest to them. Thus, this paper explores an automated way to help answer the always recurring question from domain experts *"Which approximation method is best for my data?"*. An evolutionary algorithm is presented that combines automatic model *type* selection, automatic model parameter optimization, and sequential design exploration.

In the next Section we describe the problem of global surrogate modeling followed by an in depth discussion of the motivation for this work in Section 3. The core approach presented in this paper is discussed in Section 5 followed by a critical analysis in Section 6. Section 7 describes a number of surrogate modeling problems we shall use to demonstrate the proposed approach, followed by their discussion in Section 10 (the experimental setup is described in Section 9). We conclude in Section 12 with pointers to future work.

2. Global Surrogate Modeling

The mathematical formulation of the problem is as follows: approximate an unknown multivariate function $f : \Omega \mapsto \mathbb{C}^n$, defined on some domain $\Omega \subset \mathbb{R}^d$, whose function values $f(X) = \{f(x_1), ..., f(x_k)\} \subset \mathbb{C}^n$ are known at a fixed set of pairwise distinct sample points $X = \{x_1, ..., x_k\} \subset \Omega$. Constructing an approximation requires finding a suitable function s from an approximation space S such that $s : \Omega \mapsto \mathbb{C}^n \in S$ and s closely resembles f as measured by some criterion ξ, where ξ constitutes three parts:

$$\xi = (\Lambda, \varepsilon, \tau).$$

Λ is the generalization estimator, ε the error (or loss) function, and τ is the target value required by the user. This means that the global surrogate model generation problem (i.e., finding the best approximation $s^* \in S$) for a given set of data points $D = (X, f(X))$ can be formally defined as

$$s^* = \arg\min_{t \in T} \arg\min_{\theta \in \Theta} \Lambda(\varepsilon, s_{t,\theta}, D) \tag{1}$$

such that

$$\Lambda(\varepsilon, s^*_{t,\theta}, D) \leqslant \tau$$

where $s_{t,\theta}$ is the parametrization θ (from a parameter space Θ) of s and $s_{t,\theta}$ is of model type t (from a set of model types T).

1. The terms surrogate model and metamodel are used interchangeably.

The first minimization over $t \in T$ is the task of selecting a suitable approximation model type, that is, a rational function, a neural network, a spline, etc. This is the model type selection problem. In practice, one typically considers only a single $t \in T$, though others may be included for comparison. Then given a particular approximation type t, the task is to find the hyperparameter assignment θ that minimizes the generalization estimator Λ (e.g., determine the optimal order of a polynomial model). This is the hyperparameter optimization problem, though generally both minimization's are simply referred to as the model selection problem. Many implementations of Λ have been described: the hold-out, bootstrap, cross validation, jack-knife, Akaike's Information Criterion (AIC), etc. Different criteria may also be combined in a multi-objective fashion. In this case ξ is really a matrix

$$\xi = \begin{bmatrix} \Lambda_1 & \varepsilon_1 & \tau_1 \\ \Lambda_2 & \varepsilon_2 & \tau_2 \\ \dots & \dots & \dots \\ \Lambda_m & \varepsilon_m & \tau_m \end{bmatrix}$$

with m the number of objectives. A simple example is minimizing the average relative cross validation error together with the maximum absolute deviation in the training points. An additional assumption is that f is expensive to compute. Thus the number of function evaluations $|f(X)|$ needs to be minimized and data points must be selected iteratively, at points where the information gain will be the greatest (Turner et al., 2007). Mathematically this means defining a sampling function

$$\phi(X_{j-1}) = X_j\,,\; j = 1,..,N$$

that constructs a data hierarchy

$$X_0 \subset X_1 \subset X_2 \subset ... \subset X_N \subset X$$

of nested subsets of X, where N is the number of levels. X_0 is referred to as the *initial experimental design* and is constructed using one of the many algorithms available from the theory of Design and Analysis of Computer Experiments (Kleijnen et al., 2005). Once the initial design X_0 is available it can be used to seed the sampling function ϕ. An important requirement of ϕ is to minimize the number of sample points $|X_j| - |X_{j-1}|$ selected each iteration, yet maximize the information gain of each successive data level. Depending on the problem, ϕ can take into account different criteria (non-linearity of the response, smoothness/uncertainty of the model, location of the optima, etc.). This process is referred to as adaptive sampling, active learning, model updating, or sequential design.

An important consequence of the adaptive sampling procedure is that the task of finding the best approximation s^* (cfr. Equation 1) becomes a dynamic problem instead of a static one. Since the optimal model parameters will change as the amount and distribution of data points changes. This of course makes the problem more difficult.

3. Motivation

While the mathematical formulation of global surrogate modeling is clear cut, its practical implementation raises an obvious question: How should the minimization over $t \in T$ and $\theta \in \Theta$ in Equation 1 be performed? We discuss both cases in the following subsections.

3.1 Problem 1: Model Type Selection

The first minimization over $t \in T$ is the model type selection problem. Many model types exist: rational functions, Artificial Neural Networks (ANN), Support Vector Machines (SVM), Gaussian Process (GP) models, Multivariate Adaptive Regression Splines (MARS), Radial Basis Function (RBF) models, projection pursuit regression, rational functions, etc.

3.1.1 BACKGROUND

From a theoretic standpoint, selecting the most suitable approximation method for a given response is a difficult problem that depends on the data characteristics (dimensionality, number of points, distribution, noise level, periodicity, etc) and the application constraints (accuracy, smoothness, ability to capture poles or discontinuities, execution speed, interpretability, extrapolation requirements, etc). Different application domains prefer different model types. For example rational functions are widely used by the electrical engineering community (Deschrijver and Dhaene, 2005) while ANN are preferred for hydrological modeling (Solomatine and Ostfeld, 2008). Differences in model type usage are often due to practical reasons. This is particularly true in industrial settings. For example: the designer adheres to common practice within his field, the final application restricts the designer to one particular type (e.g., rational models for EM systems), or the expertise is not available to properly try other methods.

Of course, this need not always be the case. The choice of the metamodel type can also be motivated by knowledge of the underlying physics[2] (Triverio et al., 2007) or by the special features the model provides: for example the uncertainty prediction based on random process assumption in Kriging methods[3] (Xiong et al., 2007). So remark there is no such thing as an inherently 'good' or 'bad' model. A model is only as good as the data it is based on and the expertise of the user that built it.

3.1.2 CLASSIC APPROACH

If multiple model types are considered, the classic approach is to simply to try out different types and select the best one according to one or more accuracy criteria. There is ample literature available that benchmarks model types in this way (Simpson et al., 2001; Jin et al., 2001; Queipo et al., 2005; Yang et al., 2005; Chen et al., 2006; Wang and Shan, 2007; Chung and Alonso, 2000; Gano et al., 2006; Gu, 2001; Santner et al., 2003; Lim et al., 2007; Fang et al., 2005; Gorissen et al., 2009c). But claims that a particular model type is superior to others should always be met with some skepticism.

In order for the different benchmarking studies to be truly useful for a domain expert, the results of such studies must be collected and compiled into a general set of rules, recipe, or flowchart. To ease the discussion, let us denote such a compilation into a learning algorithm by L. L is then essentially a classifier that can predict which model type $t \in T$ to use based on data D and application requirements Γ:

2. Knowledge of the physics of the underlying system can make a particular model type to be preferred. For example, rational functions are popular for all kinds of Linear Time-Invariant systems since theory is available that can be used to prove that rational pole-residue models conserve certain physical quantities (Triverio et al., 2007).
3. Kriging models are closely related to GP models and and often Kriging and GP models are used as labels for the same techniques. Great similarities between GP models, SVM models, RBF models, and RBF Neural Networks exist as well, as has been discussed in Rasmussen and Williams (2006).

$$L(D,\Gamma) = t.$$

When executed L should then be able to give a specific recommendation as to which model type to use for a given problem. This recommendation should be more specific than the general rules of thumb that are available now. Experience shows this to be exactly what an application engineer wants. However, constructing such a learner L for any but the most restricted class of problems is a daunting undertaking for obvious practical reasons. Firstly, deciding which problem/application features to train the classifier on is far from trivial. Also even if this is done, the number of features can be expected to be high thus gathering the necessary data (by manually solving Equation 1) to train L accurately will be very computationally expensive.

Secondly, as mentioned above, the success of a model type largely depends on the expertise of the user, the quality of the data, and even the quality of the software implementation of the technique. Neural networks are a good example in this respect. In the right hands they are able to perform very well on many problems. However, if poor choices are made with regard to training function, topology selection, generalization control, training parameters, software library, etc. they may seem to perform poorly. How to take into account this information in L?

Thirdly, a more fundamental problem with this approach is that data must be available in order for the reasoner to work. However, if only a simulation code is available (as is often the case) data must be collected, and the optimal data collection strategy that minimizes the number of points depends on the model type. Also, the optimal model type will change depending on how much data is available. One could argue to instead train L only on the data characteristics which are known in advance (e.g., dimensionality, noise level, etc.). The question is then again, which characteristics are most important? Furthermore, in many cases not much is known about the true behavior of the response thus there will typically not be enough information to train L accurately.

This brings us to the final point. A main reason for turning towards global surrogate modeling methods is that little is known about the behavior of the response (Simpson et al., 2008). The goal is to get insight into that behavior in a computationally cheap way by applying surrogate methods. Another reason why information about the data may be scarce is that the source of the data is confidential or proprietary and very little information is disclosed. In these situations using or training L becomes very difficult.

Finally, we must stress that we do ***not*** say that this problem is too difficult and not worth trying to solve. Indeed many such problems exist and are currently being tackled, particularly in medicine. Instead we argue that users of global surrogate modeling methods can benefit from a more dynamic approach that is flexible, can be easily applied to a wide range of different problems, can easily incorporate new fitting techniques and process knowledge, and naturally integrates with an adaptive data collection procedure. We shall revisit this point in sections 3.3 and 6.

3.2 Problem 2: Model Parameter Selection

Assuming the model type selection problem has been solved, there remains the model parameter selection problem (the minimization over $\theta \in \Theta$ in Equation 1). For example, finding the optimal C, ε and σ parameters in the case of RBF SVMs. This is the classic hyperparameter optimization problem that also depends on the data characteristics (for example the optimal correlation function and correlations parameters of a Kriging model will depend on the data distribution (Gorissen et al., 2008b; Toal et al., 2008). Some models are more sensitive to changes in their parameters than others

and usually it takes a great deal of experience to know how all parameters should be set. Sometimes this problem is solved through trial and error, but usually it is tackled as an optimization problem and classic optimization algorithms are used guided by a performance metric.

A huge amount of research has been done on this topic, particularly in the machine learning community (see Section 4). This particular problem is not the main focus of this work. Rather we are more interested in tackling the first problem.

3.3 Proposed Solution

While we are primarily interested in the first problem, the approach described in this paper naturally incorporates problem 2 as well. In both cases there is little theory that can be used as a guide. It is in this setting that the evolutionary approach can be expected to do well. We describe the application of a single GA with speciation to both problems: the selection of the surrogate type and the optimization of the surrogate model parameters (= hyperparameter optimization). In addition, we do not assume all data is available at once but must be sampled incrementally since it is expensive (active learning).

The idea is to maintain a heterogeneous population of surrogate model types and let them evolve cooperatively and dynamically with the changing data distribution. The details will be presented in Section 5 and a critique in Section 6. In addition, an implementation in the form of a Matlab toolbox is available for download from `http://www.sumo.intec.ugent.be`.

4. Related Work

The evolutionary generation of regression models for given input-output data has been widely studied in the genetic programming community (Vladislavleva et al., 2009; Streeter and Becker, 2003; Yeun et al., 2004). Given a set of mathematical primitives (+, *sin*, *exp*, /, *x*, *y*, etc.) the space of symbolic expression trees is searched to find the best function approximation. The application of GAs to the optimization of model parameters of a single model type (homogeneous evolution) has also been common (Chen et al., 2004; Lessmann et al., 2006; Tomioka et al., 2007; Friedrichs and Igel, 2005; Zhang et al., 2000) and the extensive work by Yao (1999); Yao and Xu (2006). Integration with adaptive sampling has also been discussed (Busby et al., 2007). However, these efforts do not tackle the model *type* selection problem, they restrict themselves to a particular method (e.g., SVMs or neural networks). As Knowles and Nakayama (2008) state "*Little is known about which types of model accord best with particular features of a landscape and, in any case, very little may be known to guide this choice*.". Likewise, Solomatine and Ostfeld (2008) note: "*...it is important to stress that there are always situations when one model type cannot be applied or suffers from inadequacies and can be well complemented or replaced by another one*". Thus an algorithm to help solve this problem in a dynamic, automated way is very useful (Keys et al., 2007). This is also noticed by Voutchkov and Keane (2006) who compare different surrogate models for approximating each objective during optimization. They note that in theory their approach allows the use of a different model type for each objective. However, such an approach will still require an a priori model type selection and does not allow for dynamic switching of the model type or the generation of hybrids.

There has also been much research on the use of surrogate models in evolutionary optimization of expensive simulators (to approximate the fitness function). References include Jin et al. (2002), Regis and Shoemaker (2004), Paenke et al. (2006) and Emmerich et al. (2006), the work by Ong et al. (2006), and more recently by Lim et al. (2007). In general the theory is referred to as Surrogate

Based Optimization or Metamodel Assisted Optimization. A good overview reference is given by Eldred and Dunlavy (2006) and Queipo et al. (2005). For example, Lim et al. (2007) compare the utility of different local surrogate modeling techniques (quadratic polynomials, GP, RBF, ...) including the use of (fixed) ensembles, for optimization of computationally expensive simulation codes. Local surrogates are used together with a trust region framework to quickly and robustly identify the optimum. As noted in the introduction, the contrast with this work is that references such as Lim et al. (2007) are interested in the optimum and not the surrogate itself (they make only a "*mild assumption on the accuracy of the metamodeling technique*"). In addition the model parameters are taken as fixed and there is no integration with active learning. In contrast we place very strong emphasis on the surrogate model accuracy, the automatic setting of the hyperparameters, and the efficient sampling of the *complete* design space.

The work of Sanchez et al. (2006) and Goel et al. (2007) is more useful in our context since they provide new algorithms for generating an optimal set of ensemble members for a fixed set of data points (no sampling). Unfortunately, though, the parameters of the models involved must still be chosen manually. Nevertheless, their approaches are interesting, and can be used to further enhance the approach presented here. For example, instead of returning the single final best model, an optimal ensemble member selection algorithm can be used to return a potentially much better model based on the final population or Pareto front.

From machine learning the work in B. et al. (2004) is also related. The authors describe an interesting classification algorithm *COMB* that combines online an ensemble of active learners so as to expedite the learning progress in pool-based active learning. In their terminology an active learner is a combination of a model type and a sampling algorithm. A weighted ensemble of active learners is maintained and each learner is allowed to express interest in a pool of unlabeled training points. Depending on the interests of the active learners, an unlabeled point is selected, labeled by the teacher, and based on the added value of that point the different active learners are punished or rewarded. Internally the active learners are SVM models whose parameters are chosen manually. In principle, with a number of approximations one could adapt the algorithm to the regression case. If one then also included hyperparameter optimization, the result would be very similar to the SUMO-Toolbox (cfr. Section 5.2) configured with one or more of the *Error*, LRM, or EGO (Jones et al., 1998) sample selection algorithms, but without the ability to combine different criteria. However, a problem would be that *COMB* assumes a pool of unlabeled training data is given. However, when modeling a simulation code in regression no such pool is available. Some external algorithm would still be needed to generate it in order for *COMB* to work. *COMB* does also naturally allow for different model types but in a more static way than the algorithm in Section 5.3: there is no hyperparameter optimization, the number of each active learning type remains fixed (though the weights can change) leading to a potentially high computational cost, and hybrid models are not considered. The extension to the multi-objective case is also non-trivial. Of course *COMB* could be extended to incorporate such features, but the result would be very similar to the work presented here. Nevertheless, the specific scoring functions, probability weightings, and ensemble weight updates, seem very useful and could be implemented in the SUMO-Toolbox to complement the approach presented here.

Finally, the work by Escalante et al. (2008) is most similar to the topic of this paper. Escalante et al. (2008) consider the problem of finding the optimal classifier and associated hyperparameters for a given classification problem (active learning is not considered). A solution is encoded as a vector and Particle Swarm Optimization (PSO) is used to search for good classifiers. Good results

are shown on various benchmarks. Unlike the GA approach, however, it is less straightforward to cater for multiple sub-populations, giving models room to mature independently before entering competition. The use of operators tuned to specific models is also difficult (to increase the search efficiency). In effect, the PSO approach takes a top-down view, using a high level encoding in a high dimensional space, a typical particle has 25 dimensions (Escalante et al., 2007). In contrast the GA approach is bottom up, the model specific operators result in a much smaller search space, different for each method (e.g., 1 for the spline models and 2 for the SVM models). This leads to a more efficient search requiring less fitness evaluations and facilitates the incorporation of prior knowledge. In addition, by using PSO there is no natural way of enabling hybrid solutions (ensembles) without extending the encoding and further increasing the search space. In contrast, the hybrid solutions arise very naturally in the GA framework and do not impact the search space of the other model types. The same is true of the extension to the multi-objective case, a very natural step in the GA case.

In sum, in by far the majority of the related work considered by the authors, speciation was always constrained to one particular model type, for example neural networks in Stanley and Miikkulainen (2002). The model type selection problem was still left as an a-priori choice for the user. Or, if multiple model types are used, the hyperparameters are typically kept fixed and there is no tie-in with the active learning process.

5. Heterogeneous Evolution of Surrogate Models

This Section discusses how different surrogate models may be evolved cooperatively in order perform model type selection.

5.1 Speciated Evolution

Since GAs are population-based they easily lend themselves to parallelism. The terms Parallel Genetic Algorithms (PGA) or Distributed Genetic Algorithms (DGA) refer to the case whenever the population is divided up in some way, be it to improve the computational efficiency or search efficiency. Unfortunately though, the terminology varies between authors and can be confusing (Nowostawski and Poli, 1999; Alba and Tomassini, 2002). From a biological standpoint it makes sense to consider *speciation:* genomes that differ considerably from the rest of the population are segregated and continue to evolve semi-independently, forming a new species.

The *island model* (Whitley et al., 1999.; Hocaoglu and Sanderson, 2001; Giannakoglou et al., 2006) is probably the most well known PGA. Different sub-populations, called *demes*, exist (initialized differently) and sporadic migration can occur between islands allowing for the exchange of genetic material between species and inter-species competition for resources. Selection and recombination are restricted per deme, such that each sub-population may evolve towards different locally optimal regions of the search space (also called *niches*). An advantage of using migration is that it allows sub-species to mature in semi-isolation without being forced to consistently engage in competition. This is particularly useful for the application of this paper. The island model introduces five new parameters: the migration topology, the migration frequency, the number of individuals to migrate, a strategy to select the emigrants, and a replacement strategy to incorporate the immigrants. The island model is illustrated in Figure 1 for two topologies.

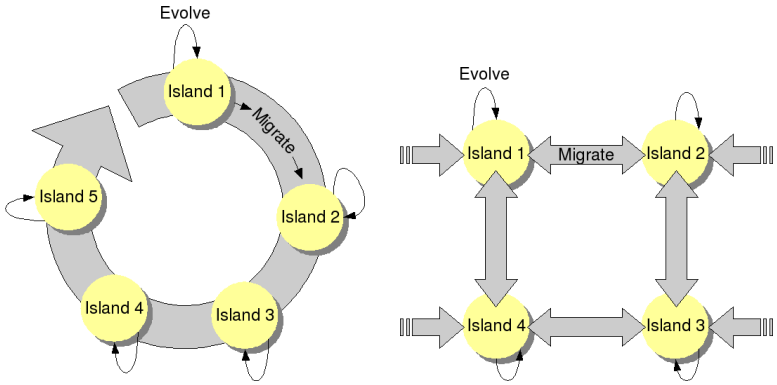


Figure 1: Ring (left) and grid (right) migration topologies in the Island Model

5.2 Global Surrogate Modeling Control Flow

Before we can discuss the concrete implementation of the automatic model type selection algorithm it is important to revisit the general global surrogate modeling methodology described in Section 2. It is important to understand the general control flow since it forms the basis for the evolutionary algorithm described in the next Section.

The general methodology is as follows: Initially, a small initial set of samples is chosen according to some experimental design (e.g., Latin hypercube, Box-Behnken, etc.). Based on this initial set, one or more surrogate models are constructed and their hyperparameters optimized according to a chosen hyperparameter optimization algorithm (e.g., BFGS, Particle Swarm Optimization (PSO), Genetic Algorithm (GA), DIRECT, NSGA-II, etc.). Models are assigned a score based on one or more measures (e.g., cross validation, Akaike's Information Criterion (AIC), etc.) and the optimization continues until no further improvement is possible. The models are then ranked according to their score and new samples are selected based on the best performing models and the behavior of the response (the exact criteria depend on the active learning algorithm used). The hyperparameter optimization process is continued or restarted intelligently and the whole process repeats itself until one of the following three conditions is satisfied: (1) the maximum number of samples has been reached, (2) the maximum allowed time has been exceeded, or (3) the user required accuracy has been met.

Recall that the adaptive sampling procedure has an important effect on the hyperparameter optimization. The non-stationary data distribution makes the model parameter optimization surface dynamic instead of static (as is typically assumed).

A readily available implementation of the control flow described in this Section, and the one we shall use for the experiments in this paper, is available as the **SU**rrogate **MO**deling Toolbox (SUMO Toolbox) (Gorissen et al., 2009c) from `http://www.sumo.intec.ugent.be`.

5.3 Algorithm

Algorithm 1 Automatic global surrogate modeling with heterogeneous evolution and active learning

01. $X_0 = initialExperimentalDesign()$;
02. $X = X_0$;
03 . $f|_X = evaluateSamples(X)$;
04 .$T = \{t_1, ..., t_h\}$;
05. $M_i = \text{createInitialModels}(t_i, popsize_i); i = 1, .., h$
06. $M = \bigcup_{i=1}^{h} M_i$;

07 . **while** (ξ not reached) **do**
08. $scores = \{\}$; $gen = 1$;
09 . **while** ($termination\_criteria$ not reached) **do**
10. **foreach** $M_i \subseteq M$ **do**
11. $scores_i = \text{fitness}(M_i, X, f|_X, \xi)$;
12. $elite = \text{sort}([scores_i; M_i])|_{1:el}$;
13. $parents = \text{select}(scores_i, M_i)$;
14. $parents_{xo} = \text{selectXOParents}(parents, p_c)$;
15. $offspring_{xo} = \text{crossover}(parents_{xo}, ES_{diff}, ES_{max})$;
16. $parents_{mut} = parents \backslash parents_{xo}$;
17. $offspring_{mut} = \text{mutate}(parents_{mut})$;
18. $M_i = elite \bigcup offspring_{mut} \bigcup offspring_{xo}$;
19. $scores = scores_i \cup scores$;
20. **end**
21. **if** $(mod(gen, m_i) = 0)$
22. $M = \text{migrate}(M, scores, m_f, m_d)$
23. **end**
24. $M = \text{extinctionPrevention}(M, T_{min})$;
25. $gen = gen + 1$;
26. **end**
27 . $X_{new} = \text{selectSamples}(X, f|_X, M)$;
28 . $f|_{X_{new}} = \text{evaluateSamples}(X_{new})$;
29 . $[X, f|_X] = merge(X, f|_X, X_{new}, f|_{X_{new}})$;
30 . **end**

31 . **return**bestModel(M);

We now present the concrete GA for heterogeneous evolution as it is embedded (as a plugin) in the SUMO Toolbox. The speciation model used is the island model since we found it the most natural way of evolving multiple model types while still allowing for hybrid solutions. The algorithm is based on the Matlab GADS toolbox and works as follows (see Algorithm 1 and reference (Gorissen, 2007) for more details): After the initial DOE has been calculated (cfr. the control flow in Section 5.2), an initial sub-population M_i is created for each model type $t \in T$ ($i = 1, .., h$). The exact creation algorithm is different for each model type so that model specific knowledge can be exploited.

Subsequently, each deme is allowed to evolve according to an elitist GA. Parents are selected according a selection algorithm (e.g., tournament selection) and offspring undergo either crossover (with probability p_c) or mutation (with probability $1 - p_c$). The models M_i are implemented as Matlab objects (with full polymorphism) thus each model type can choose its own representation and mutation/crossover implementations (this implements the minimization over $\theta \in \Theta$ of Equation 1). While mutation is straightforward, the crossover operator is more involved (see Section 5.5 below).

The fitness function calculates the quality of the model fit, according to criteria ξ. The current deme population is then replaced with its offspring together with *el* elite individuals. Once every deme has gone through a generation, migration between individuals is allowed to occur at migration interval m_i, with migration fraction m_f and migration direction m_d (a ring topology is used). The migration strategy is as follows: the $l = (|M_i| \cdot m_f)$ fittest individuals of M_i replace the l worst individuals in the next deme (defined by m_d). As in Pei and Goodman (2001), migrants are duplicated, not removed from the source population. Note that in this contribution we are primarily concerned with inter-model speciation (speciation as in different model types). Intra-model speciation (e.g., through the use of fitness sharing within one model type) is something which was not done but could easily be incorporated.

Once the GA has terminated, control passes back to the main global surrogate modeling algorithm of the SUMO Toolbox. At that point M contains the best set of models that can be constructed for the given data. If the accuracy of the models is sufficient the main loop terminates. If not, a new set of maximally informative sample points is selected based on several criteria (quality of the models, non-linearity of the response, etc.) and scheduled for evaluation. Once new simulations become available the GA is resumed.

Note that sample evaluation and model construction/hyperparameter optimization run in parallel. For clarity, algorithm 1 shows them running sequentially but this is not what happens in practice. In reality both are interleaved to allow an optimal use of computational resources.

5.4 Extinction Prevention

Initial versions of this algorithm exposed a major shortcoming, specifically due to the fact that models are being evolved. Since not all data is available at once but trickles in, $|X_j| - |X_{j-1}|$ samples at a time, models that need a reasonable-to-large number of samples to work well will be at a huge disadvantage initially. Since they perform badly at first, they may get overwhelmed by other models who are less sensitive to this problem. In the extreme case where they are driven extinct, they will never have had a fair chance to compete when sufficient data *does* become available. They may even have been the superior choice had they still been around.[4] Therefore an Extinction Prevention (EP) algorithm was introduced that ensures a model type can never disappear completely.

EP works by monitoring the population and each generation recording the number of individuals of each model type. If this number falls below a certain threshold T_{min} for a certain model type, the EP algorithm steps in and ensures the model type has its numbers replenished up to the threshold. This is done by re-inserting the last models that disappeared for that type (making copies if necessary). The re-inserted models replace the worst individuals of the other model types (who do have sufficient numbers) evenly.

Strictly speaking, EP goes completely against the survival of the fittest principle in evolutionary algorithms. By using it we are manually working against selection, preserving model types which

4. As an example, this observation was often made when using rational models on electro-magnetic data.

give poor results at that point in time. However, in this setting it seems a fair measure to take (we do not want to risk loosing a model type completely) and improves results in most cases (*see* Section 10). At the same time it is straightforward to implement and understand, needing no special control parameters. All it has to ensure is that a species is never driven extinct.

5.5 Heterogeneous Recombination

The attentive reader will have noticed that one major problem remains with the implementation as discussed so far. The problem lies in the genetic operators, more specifically in the crossover operator. Migration between demes means that model types will mix. This means that a set of parents selected for reproduction may contain more than one model type. The question then arises: how to perform recombination between two models of completely different types. For example, how to meaningfully cross an Artificial Neural Network with a rational function? The solution we propose here is to use ensembles (behavioral recombination). If two models of different types are selected to recombine, an ensemble is created with the models as ensemble members. Thus, as soon as migration occurs, model types start mixing, and ensemble models arise as a result. These are treated as a distinct model type just as the other model types.

However, the danger with this approach is that the population may quickly be overwhelmed by large ensembles containing duplicates of the best models (as was noticed during initial tests). To counter this phenomenon we apply the similarity idea from Holland's sharing concept (Holland, 1975). Individual models will try to mate only with individuals of the same type. Only in the case where selection has made this impossible shall different model types combine to form an ensemble. In addition we enforce a maximum ensemble size ES_{max} and require that ensemble members must differ ES_{diff} percent in their response (their 'behavior'). This is calculated by evaluating the models on a dense grid.

This leaves us with only three cases left to explain:

1. *ensemble - ensemble* recombination: a single-point crossover is made between the ensemble member lists of each model (note that the type of the ensemble members is irrelevant)

2. *ensemble - model* recombination: the model replaces a randomly selected ensemble member with probability p_{swap} or gets absorbed into the ensemble with probability $1 - p_{swap}$ (respecting ES_{max} and ES_{diff}).

3. *ensemble* mutation: one ensemble member is randomly deleted

Besides enabling hybrid solutions, using ensembles has the additional benefit of allowing a model to lie 'dormant' in an ensemble with the possibility of re-emerging later (e.g., if after mutation only one ensemble member remains). Note that, in contrast to Lim et al. (2007) for example, the type of the ensemble members is not fixed in any way but varies dynamically.

We have not yet mentioned what type of ensemble will be used. There are several methods for combining the outputs of models, such as average, weighted average, Dempster-Shafer methods, using rank-based information, supra-Bayesian approach, stacked generalization, etc (Sharkey, 1996). To keep the implementation straightforward and the complexity (number of parameters) low we have opted for a simple average ensemble. Of course different, more powerful combination methods could be used instead and they will only improve results. The exact method used is of lesser importance since it does not change the methodology. The advantage of a simple average

ensemble is that it works in all cases: It makes no assumption on the model types involved, nor does it mandate any changes to the models or training algorithms (for example, like negative correlation learning) since this is not always possible (e.g., when using proprietary, application specific, modeling code).

5.6 Multi-objective Model Selection

A crucial aspect of the model generation algorithm is the choice of a suitable criteria ξ. In practice it turns out that selecting an appropriate function for Λ, ε and a target value for τ is difficult. Particularly if little is known about the structure of the response. This is related to the "*The 5 percent problem*" (Gorissen et al., 2009b). The fundamental reason for this difficulty is that an approximation task inherently involves multiple, conflicting, criteria (Li and Zhao, 2006). Thus a multi-objective approach is very useful here since it enables the use of multiple criteria during the hyperparameter optimization process (see Jin and Sendhoff 2008 for an excellent overview of this line of research).

Secondly, it is not uncommon that a simulation engine has multiple outputs that all need to be modeled (Conti and O'Hagan, 2007). The direct approach is to model each output independently with separate models (possibly sharing the same data). This, however, leaves no room for trade-offs nor gives any information about the correlation between different outputs. Instead of performing two modeling runs (doing a separate hyperparameter optimization for each output) both outputs can be modeled simultaneously if models with multiple outputs are used in conjunction with a multi-objective optimization routine.

In both cases such a multi-objective approach can be integrated with the automatic surrogate model type selection algorithm described here. This means that the best model type can vary per criteria or, more interestingly, that it enables automatic selection of the best model type for each output without having to resort to multiple runs. A full discussion of these topics is out of scope for this paper. However, details and some initial results can already be found in Gorissen et al. (2009a) and Gorissen et al. (2009b).

6. Critique

The algorithm presented so far has a number of strengths and weaknesses. The obvious advantage is the ability to perform automatic selection of the model type and complexity for a given data source (no need to do multiple parallel runs or train a complex classifier). In addition the algorithm is generic in that it is independent of the data origin (application), model type, and data collection strategy. New approximation methods can easily be incorporated without changing the algorithm. Problem specific knowledge and model type specific optimizations based on expert knowledge can also be incorporated if needed (i.e., by customizing the genetic operators). Furthermore, the algorithm naturally integrates with the data collection strategy, allowing the best model type to change dynamically and naturally allows for hybrid solutions. Finally, it naturally extends to the multi-objective case (Section 5.6) and can be easily parallelized to allow for faster computations (though the computational cost is still outweighed by the simulation cost).

The main disadvantage is due to the fact that the approach is based on evolutionary algorithms: full determinism can not be guaranteed. This raises the obvious question of how stable the convergence is over multiple runs. The same can be said of standard approaches towards hyperparameter optimization (which typically include randomization) or for any algorithm involving a GA for that

matter. Formulating theoretical foundations in order to come to convergence guarantees for GAs is a difficult undertaking and has been the topic of intense research ever since their inception in the late 80s. Characterizing the performance of genetic algorithms is complex and depends on the application domain as well as the implementation parameters (Rawlins, 1991). Most theoretic work has been done on schema theorems for the Canonical Genetic Algorithm (CGA), which try to prove convergence in a simplified framework using a binary representation. However prediction of the future behavior of a GA turns out to be very difficult and much controversy remains over the usefulness of these theorems (Poli, 2001; Goldberg, 1989). Theoretical work on other classes of GAs or using specific operators has also been done (Nakama, 2008; Neubauer, 1997; Rawlins, 1991; Qi and Palmieri, 1994a,b) but is unfortunately of little practical use here. For example, the work in Ankenbrandt (1990) requires the calculation of fitness ratios, but this is impractical (and computationally expensive) to do in this situation and the results will vary with the application.

Thus, for the purposes of this paper a full mathematical treatment of algorithm 1 and its convergence is out of scope. Due to the island model, sampling procedure, and heterogeneous representation/operators used, such a treatment will be far from trivial to construct and distract from the main theme of the paper. In addition its practical usefulness would remain questionable due to the many assumptions that will be required. However, gaining a deeper theoretical insight into the robustness of the algorithm is still very important. A sensitivity study of the main GA parameters involved will shed more light on this issue.

Theoretical remarks aside, the authors have found that in practice the approach works quite well. If reasonable population sizes are used together with migration and the extinction prevention algorithm described in Section 5.4, the results of the algorithm are quite robust and give useful results and insights into the modeling problem. Besides the results given in this paper, good results have also been reported on various real world problems from aerodynamics (Gorissen et al., 2009a), electronics (Gorissen et al., 2008a), hydrology (Couckuyt et al., 2009), and chemistry (Gorissen, 2007).

In sum, this approach is useful if: little information is known about the expected structure of the response, if it is unclear which model type is most suited to the problem, data is expensive and must be collected iteratively, and hybrid solutions are useful. In other cases, for example it is clear from a priori knowledge which model type will be the most suitable (e.g., based on existing rules of thumb for a well defined, restricted problem), this approach should not be applied, save as a comparison.

7. Test Problems

We now consider five test problems to which we apply the heterogeneous GA (from now on abbreviated by HGA). The objective is to validate if the best model type can indeed be determined automatically, and in a way that is cheaper and better than the simple brute force method: doing multiple, single model type runs in parallel. The problems include 2 predefined mathematical functions, and real-world problems from electronics and aerodynamics.

The dimensionality of the examples ranges from 2 to 13. This is no inherent limit but simply depends on the model types used. For example if only SVM-type models are used the number of dimensions can be arbitrarily high, while for smoothing spline models the dimensionality should be kept low. It all depends on which model types make up the population.

We also hope to see evidence of a 'battle' between model types. While initially one species may have the upper hand, as more data becomes available (dynamically changing hyperparameter optimization landscape) a different species may become dominant. This should result in clearly noticeable population dynamics, a kind of oscillatory stage before convergence. We briefly discuss each of the test problems in turn.

7.1 Ackley Function (AF)

The first test problem is Ackley's Path, a well known benchmark problem from optimization. Its mathematical definition for d dimensions is:

$$\begin{aligned} F(\vec{x}) &= -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{d}\cdot\sum_{i=1}^{d} x_i^2}\right) \\ &\quad -\exp\left(\frac{1}{d}\cdot\sum_{i=1}^{d}\cos(2\pi\cdot x_i)\right)+20+e \end{aligned}$$

with $x_i \in [-2,2]$ for $i = 1,...,d$. For easy visualization we take $d = 2$. For this function a validation set and a test set of 5000 random points each is available. Although this is a function from optimization we are not interested in optimizing it, rather in reproducing it using a regression method with minimal data.

7.2 Kotanchek Function (KF)

The second predefined function is the Kotanchek function (Smits and Kotanchek, 2004). Its mathematical definition is given as:

$$F(x_1,x_2,u_1,u_2,u_3) = \frac{e^{-x_2^2}}{1.2+x_1^2}+\varepsilon$$

with $x_1 \in [-2.5,1.5]$, $x_2 \in [-1.0,3.0]$, and with ε uniform random simulated numeric noise with mean 0 and variance 10^{-4}. As you can see only the first two variables are relevant. For this function a validation set and a test set of 5000 scattered points each is available.

7.3 EM Example (EE)

The fourth example is a 3D Electro-Magnetic (EM) simulator problem (Lehmensiek, 2001). Two perfectly conducting round posts, centered in the E-plane of a rectangular waveguide, are modeled, as shown in Figure 2. The 3 inputs to the simulation code are: the signal frequency f, the diameter of the posts d, and the distance between the two posts w. The outputs are the complex reflection and transmission coefficients S_{11} and S_{21}. The simulation model was constructed for a standard WR90 rectangular waveguide with $f \in$[7 GHz, 13 GHz], $d \in$[1 mm, 5 mm] and $w \in$[4 mm, 18 mm]. In addition, a 25^3 data set is available for testing purposes.

7.4 LGBB Example (LE)

NASA's Langley Research Center is developing a small launch vehicle (SLV) (Pamadi et al., 2004; Rogers et al., 2003) that can be used for rapid deployment of small payloads to low earth orbit

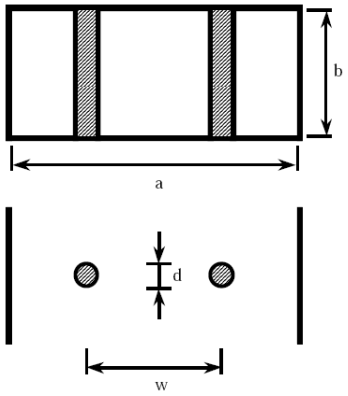
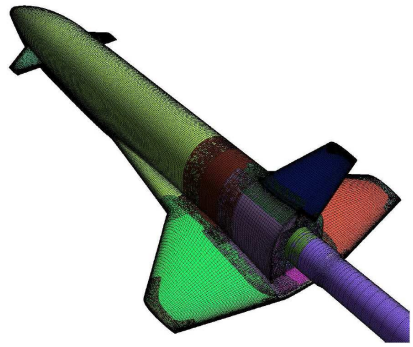
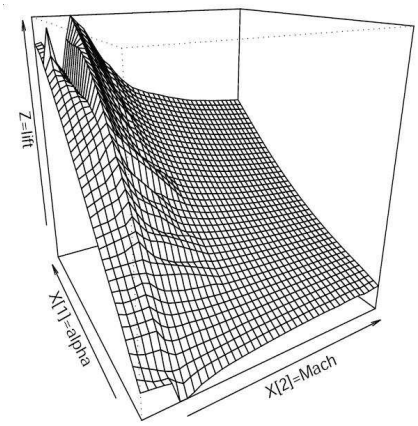


Figure 2: Cross sectional view and top view of the inductive posts (Lehmensiek, 2001)



(a) LGBB geometry (Rogers et al., 2003)



(b) Lift plotted as a function of speed and angle of attack with side-slip angle fixed to zero (Gramacy et al., 2004).

Figure 3: LGBB Example

at significantly lower launch costs, improved reliability and maintainability. The vehicle is a three-stage system with a reusable first stage and expendable upper stages. The reusable first stage booster, which glides back to launch site after staging around Mach 3 is named the Langley Glide-Back Booster (LGBB). In particular, NASA is interested in the aerodynamic characteristics of the LGBB from subsonic to supersonic speeds when the vehicle reenters the atmosphere during its gliding phase.

More concretely, the goal is to gain insight about the response in lift, drag, pitch, side-force, yaw, and roll of the LGBB as a function of three inputs: Mach number, angle of attack, and side slip angle. For each of these input configurations the Cart3D flow solver is used to solve the inviscid Euler equations over an unstructured mesh of 1.4 million cells. Each run of the Euler solver takes on the order of 5-20 hours on a high end workstation (Rogers et al., 2003). The geometry of the LGBB used in the experiments is shown in Figure 3a.

Figure 3b shows the lift response plotted as a function of speed (Mach) and angle of attack (alpha) with the side-slip angle (beta) fixed at zero. The ridge at Mach 1 separates subsonic from supersonic cases. From the figure it can be seen there is a marked phase transition between flows at subsonic and supersonic speeds. This transition is distinctly non-linear and may even be non-differentiable or non-continuous (Gramacy et al., 2004). Given the computational cost of the CFD solvers, the LGBB example is an ideal application for metamodeling techniques. Unfortunately access to the original simulation code is restricted. Instead a data set of 780 points chosen adaptively according to the method described in Gramacy et al. (2004) was used.

7.5 Boston Housing Example (BH)

The Boston Housing data set contains census information for 506 housing tracts in the Boston area and is a classic data set used in statistical analysis. It was collected by Harrison et al. and described in Harrison and Rubinfeld (1978). In the case of regression the objective is to predict the Median value of owner-occupied homes (in $1000's) from 13 input variables (e.g., per capita crime rate by town, nitric oxide concentration, pupil-teacher ratio by town, etc.).

8. Model Types

For the tests the following model types are used: Artificial Neural Networks (ANN), rational functions, RBF models, Kriging models, LS-SVMs, and for the AF example: also smoothing splines. For the EM example only the model types that support complex valued outputs directly (rational functions, RBF, Kriging) were included. Each type has its own representation and genetic operator implementation (thanks to the polymorphism as a result of the object oriented design). As stated in subsection 5.5 the result of a heterogeneous recombination will be an averaged ensemble. So in total up to seven model types will be competing to approximate the data. Remember that all model parameters are chosen automatically as part of the GA. No user input is required, the models and data points are generated automatically.

The ANN models are based on the Matlab Neural Network Toolbox and are trained with Levenberg Marquard backpropagation with Bayesian regularization (MacKay; Foresee and Hagan, 1997) (300 epochs). The topology and initial weights are determined by the GA. When run alone (without the HGA) this results in high quality models with a much faster run time than training the weights by evolution as well. Nevertheless, the high level Matlab code and complex training function do make the ANNs much slower than any of the other model types.

The LS-SVM models are based on the implementation from Suykens et al. (2002), the kernel type is fixed to RBF, leaving c and σ to be chosen by the GA. The Kriging model implementation is based on Lophaven et al. (2002) (except for the EM example) and the correlation parameters are set by the GA (the regression function is set to linear and the correlation function to Gaussian). The RBF models (and the Kriging models for the EM example, since the data is complex valued) are based on a custom implementation where the regression function, correlation function, and correlation parameters are all evolved. The rational functions are also based on a custom implementation, the free parameters being the orders of the two polynomials, the weights of each parameter, and which parameters belong in the denominator. The spline models are based on the Matlab Splines Toolbox and only have one free parameter: the smoothness.

Remember that the specific model types chosen for the different tests is less important. This can be freely chosen by the user. What is important is rather how these different model types are used

together in a single algorithm. Thus a full explanation of the virtues of each model types, as well as the representation and genetic operators used is out of scope for this paper and would consume too much space. Details can be found in Gorissen (2007) or in the implementation that is available as part of the SUMO Toolbox.

9. Experimental Setup

The following subsections describe the configuration settings used (and their motivation) for performing the experiments.

9.1 Sample Selection Settings

For the LE and BH examples only a fixed, small size, data set is available. Thus, selecting samples adaptively makes little sense. So for these examples the adaptive sampling loop was switched off. For the other examples the settings were as follows: an initial optimized Latin hypercube design, using the method from Ye et al. (2000), of size 50 is used augmented with the corner points. Modeling is allowed to commence once at least 90% of the initial samples are available. Each iteration a maximum of 50 new samples are selected using the Local Linear (LOLA) adaptive sampling algorithm (Crombecq et al., 2009). LOLA identifies new sample locations by making a tradeoff between eploration (covering the design space evenly) and exploitation (concentrating on regions where true response is nonlinear). LOLA's strengths are that it scales well with the number of dimensions, makes no assumptions about the underlying problem or surrogate model type, and works in both the $\mathbb{R}$ and $\mathbb{C}$ domains. LOLA is able to automatically identify non-linear regions in the domain and sample these more densely compared to more linear, 'flatter' regions.

By default LOLA does not rely on the (possibly misleading) approximation model, but only on the true response. This is useful here since it allows us to consider the model selection results independent from the sample selection settings. I.e., the final distribution of points chosen by LOLA is the same across all runs and model types. This means that any difference in performance between models can not be due to differences in sample distribution. However, in many cases it may be desirable to also include information about the surrogate model itself when choosing potential sample locations. In this case the LOLA algorithm can be combined with one or more other sampling criteria that do depend on model characteristics (for example the *Error*, LRM, and EGO algorithms available in SUMO).

9.2 GA Settings

The GA is run for a maximum of 15 generations between each sampling iteration (after sampling, the GA continues with the final population of the previous iteration). It terminates if one of the following conditions is satisfied: (1) the maximum number of generations is reached, or (2) 8 generations with no improvement. The size of each deme is set to 15. The migration interval m_i is set to 7, the migration fraction m_f to 0.1 and the migration direction is *both* (copies of the m_f best individuals from island i replace the worst individuals in islands $i-1$ *and* $i+1$). A stochastic uniform selection function was used. Since we want to find the best approximation over the *complete* design space, the fitness of an individual is defined as the root relative square error (RRSE):

$$RRSE(y_i, \tilde{y}_i) = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where $y_i, \tilde{y}_i, \bar{y}$ are the true, predicted, and mean true response values respectively. Intuitively the *RRSE* indicates how much better an approximation is than the most simple approximation possible (the mean) (Ganser et al., 2007). In the case of the BH and LE examples no separate validation set is available, instead 20% of the available data is reserved for this purpose (taking care to ensure the validation set is representative of the full data set by maximizing the minimum distance between validation points). Note that we are using a validation set since it is cheap and we have enough data available. In the case where data is scarce we would most likely use the more expensive k-fold cross validation as a fitness measure. This is the case for the EM example.

The remaining parameters are set as follows: $p_m = 0.2, p_c = 0.7, k = 1, p_{swap} = 0.8, el = 1, ES_{max} = 3, ES_{diff} = 0.1, T_{min} = 2$. The random generator seed was set to Matlab's default initial seed.

9.3 Termination Criteria

In case of adaptive modeling only (no sample selection), the objective is to see what the most accurate model is that can be found in a limited period of time (= a typical use case). Thus the required accuracy (= target fitness value) is set to 0. For the LE the timeout is set to 180 minutes. For the BH example the timeout is significantly extended to 1200 minutes. Given the high dimensionality, the noise and discontinuities in the input domain it is a hard problem to fit accurately. In this case we are more interested to see how the population would evolve over such an extended period of time.

In case of adaptive sampling, the criteria are: a target accuracy (RRSE) of 0.01, and for the AF example a maximum number of 500 data points is enforced (to see what performance can be reached with a limited sample budget).

9.4 Others

Each problem was modeled twice with the heterogeneous evolution algorithm (once with $EP = true$, once with $EP = false$) and once with homogeneous evolution (a single model type run for each model type in the HGA). To smooth out random effects each run was repeated 15 times. This resulted in a total of 516 runs which used up a total of at least 130 days worth of CPU time (excluding initial tests and failed runs). All experiments were run on CalcUA, the cluster available at the University of Antwerp, which consists of 256 Sun Fire V20z nodes (dual AMD Opteron with 4 or 8 GB RAM), running SUSE linux, and Matlab 7.6 R2008a. Due to space considerations, only the results for the *S11* (EE), and *lift* (LE) outputs are considered in this paper. For all examples the input space is normalized to the interval $[-1, 1]$.

10. Discussion

We now discuss the results of each problem separately in the following subsections.

10.1 Ackley Function

The composition of the final population for each run is shown in Figure 4 for Extinction Prevention (EP) equal to *true* and *EP=false*. The title above each sub figure shows the average and standard

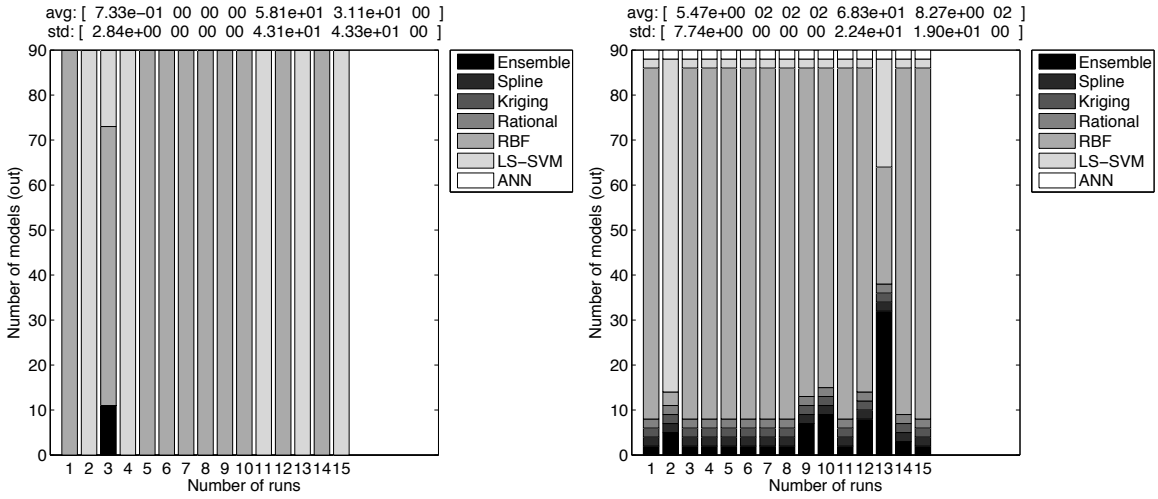


Figure 4: AF: Composition of the final population (Left: *EP=false*, Right: *EP=true*)

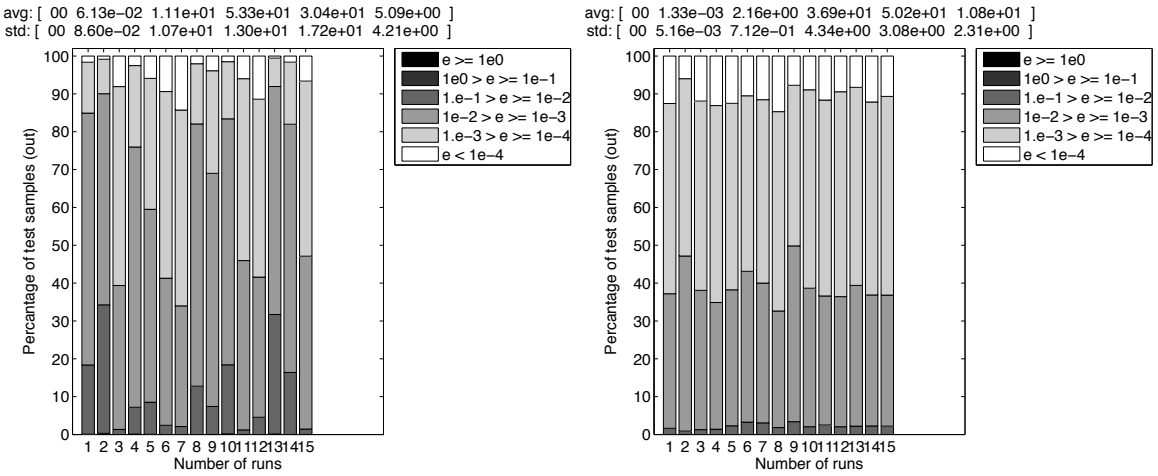


Figure 5: AF: Error histogram of the final best model in each run (Left: *EP=false*, Right: *EP=true*)

deviation over all runs. The first element of each vector corresponds to the first (top) legend entry. The error histogram of the final model of each run on the test data is shown in Figure 5. The population evolution for the run that produced the best model in both cases is shown in Figure 6. Figure 7 then depicts the evolution of the relative error (calculated according to Equation 2) on the test set as modeling progresses (again in both cases, for the run that produced the best model). The lighter the regions in Figure 7, the larger the percentage of test samples that have low relative error (RE) (according to Equation 2).

$$RE(y,\tilde{y}) = \frac{|y-\tilde{y}|}{1+|y|}. \tag{2}$$

Finally, a summary of the results for each run is shown in Table 1. The table shows the number of samples used ($|X|$), the validation error (*VE*), the test set error (*TE*), and the run time for each

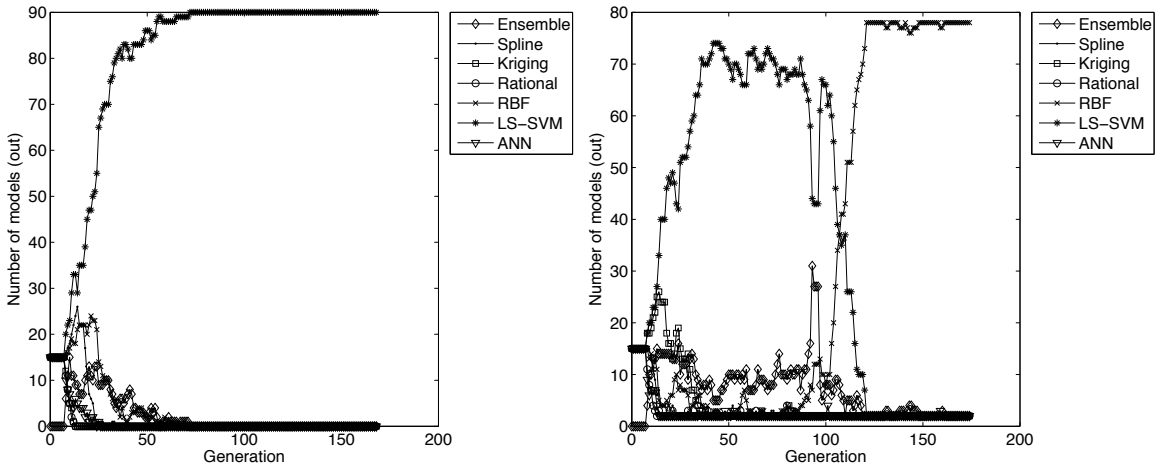


Figure 6: AF: Population evolution of the best run (Left: *EP=false*, Right: *EP=true*)

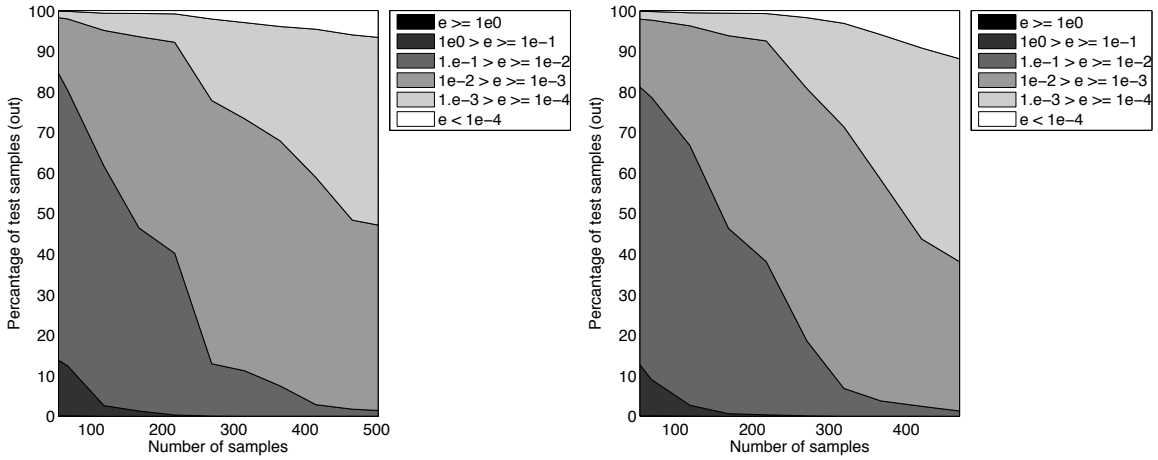


Figure 7: AF: Error evolution of the best run (Left: *EP=false*, Right: *EP=true*)

Method	$\|X\|$	σ	$\bar{VE}_{RRSE}$	σ	$\bar{TE}_{RRSE}$	σ	$\bar{time}$ *(min)*	σ
ANN	4.973E+02	1.759E+01	1.308E-02	3.428E-03	1.298E-02	3.456E-03	1.810E+02	3.176E+01
Kriging	5.263E+02	1.193E+01	2.014E-02	4.266E-03	2.038E-02	4.521E-03	7.256E+01	9.194E+00
LS-SVM	5.202E+02	1.200E+01	1.367E-02	2.262E-03	1.375E-02	2.314E-03	3.995E+01	3.276E+00
Rational	5.170E+02	1.023E+01	1.881E-01	5.854E-02	1.861E-01	5.932E-02	2.033E+01	2.221E+00
RBF	5.193E+02	1.540E+01	1.326E-02	2.365E-03	1.324E-02	2.313E-03	4.371E+01	4.133E+00
Splines	5.308E+02	1.360E+01	2.471E-02	5.660E-03	2.428E-02	5.404E-03	4.292E+01	4.984E+00
$HGA_{EP=false}$	5.055E+02	6.512E+00	3.142E-02	1.777E-02	3.094E-02	1.711E-02	2.366E+02	1.566E+02
$HGA_{EP=true}$	5.040E+02	0.000E+00	1.346E-02	1.936E-03	1.367E-02	1.897E-03	3.696E+02	6.175E+01

Table 1: AF: Comparison with homogeneous evolution

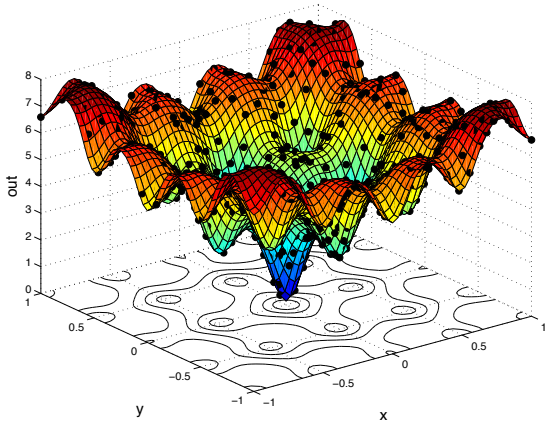


Figure 8: AF: normalized plot of the best model overall ($HGA_{EP=true}$)

experiment. All entries are averaged over 15 runs with the standard deviation shown in the adjacent column. The plot of the best model found overall is shown in Figure 8.

Regarding the composition of the final population in Figure 4, we see that the results are somewhat mixed for *EP=false*. In some runs RBF models perform best, in others LS-SVM models. This is also reflected in the corresponding error histogram plot in Figure 5. The quality of the best model found in each run differs considerably between runs. In contrast, for *EP=true*, the results are more clear cut, RBF models dominate in 14 of the 15 runs. This already demonstrates the usefulness of extinction prevention. Due to randomness in the initial population and genetic operators a model type may be driven extinct, unable to return. EP prevents this. In this particular case LS-SVM models generally perform best initially, pushing the RBF models out of the population. However, as more data becomes available (active learning), and as the hyperparameter optimization continues, superior RBF models are discovered and quickly take over the population. This is also nicely shown in Figure 6. In both cases the RBF models are driven out of the population around generation 50. Though in the *EP=true* case the RBF models are able to make a re-appearance around generation 100.

Of course nothing prevents this process from recurring. The fact that the optimal solution changes with time is not a disadvantage and should actually be expected since the optimization landscape is dynamic (due to the incremental sampling). Without EP these oscillations are impossible and everything depends on the initial conditions. As a result the danger of converging to a poor local optimum is considerably greater. Given the form of the Ackley function, we should really not be surprised that the RBF models end on top. The different radial basis functions that make up the RBF model (= a local model) can be expected to match up quite well with the 'bumps' of the Ackley function.

If we assess the quality of the final models (Figure 7) we see that it performs very well. After 500 samples the model has an error smaller than 0.01 on 98% of the test samples. More importantly, these results are consistent as can be seen from the *EP=true* plot in Figure 5. Actually, from an application standpoint consistency at this level (accuracy) is more important than consistency in model type selection. Since at the end of the day, from an application perspective, the accuracy of the model is typically most important, not its type.

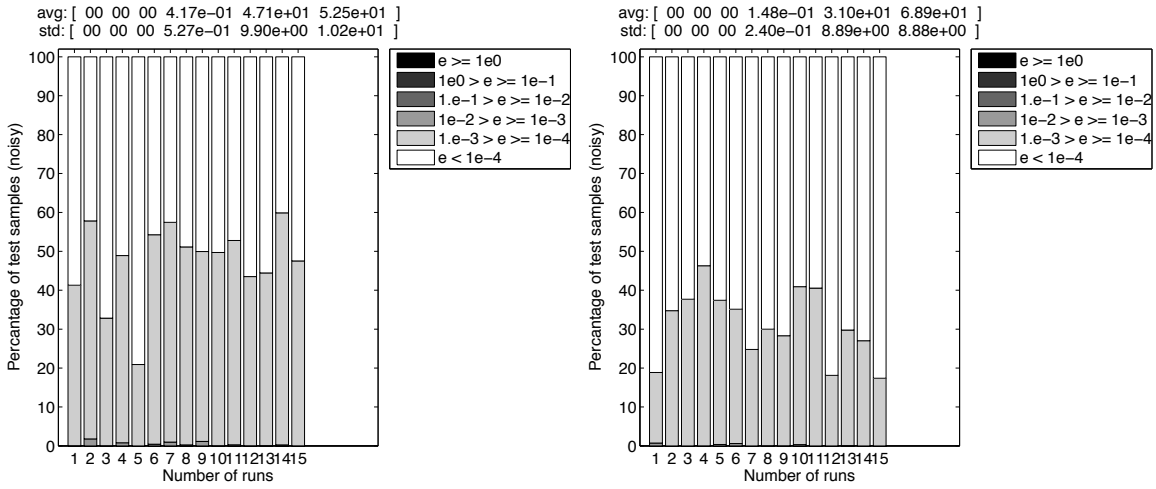


Figure 9: KF: Error histogram of the final best model in each run (Left: *EP=false*, Right: *EP=true*)

Method	$\|X\|$	σ	$\bar{VE}_{RRSE}$	σ	$\bar{TE}_{RRSE}$	σ	$\bar{time}$ *(min)*	σ
ANN	1.413E+02	2.200E+01	8.102E-04	1.614E-04	8.456E-04	1.671E-04	6.055E+01	1.258E+01
Kriging	5.200E+02	0.000E+00	1.989E-03	3.541E-04	2.023E-03	3.619E-04	1.170E+02	6.224E+01
LS-SVM	5.106E+02	2.849E+00	1.187E-01	4.967E-03	1.194E-01	5.304E-03	5.674E+01	3.571E+00
Rational	1.478E+02	9.146E+01	5.880E-04	2.115E-04	6.276E-04	2.469E-04	1.191E+01	7.547E+00
RBF	5.200E+02	0.000E+00	1.072E-01	3.620E-03	1.089E-01	4.410E-03	9.187E+01	2.157E+01
$HGA_{EP=false}$	3.059E+02	2.071E+02	1.071E-03	3.396E-04	1.085E-03	3.253E-04	3.034E+02	2.656E+02
$HGA_{EP=true}$	6.267E+01	1.486E+01	7.009E-04	2.288E-04	7.096E-04	2.024E-04	5.490E+01	3.533E+01

Table 2: KF: Comparison with homogeneous evolution

The natural question that remains, is how do these results compare with simply doing multiple homogeneous evolution (single model type, using the same GA settings) runs, one for each type separately? Those results are shown in Table 1. Studying the table we see that the HGA compares favorably. The accuracy of the final models are the essentially the same as those found by the best performing single model type run, while the variance on the results tends to be lower (EP=true). Of course this is paid for by an increase in computation time due to the increased population size of the HGA. Still, the HGA has a factor of 6 larger population size (90 vs 15) but requires only double the running time of the best performing homogeneous run (ANN). Also the total HGA running time is still less than the combined run time of all homogeneous runs.

10.2 Kotanchek Function

The composition of the final population and final error histograms for each run are shown in Figures 9 and 10. The population evolution and corresponding error evolution for the best run are shown in Figures 11 and 12. The comparison with homogeneous evolution is shown in Table 2.

The Kotanchek function is an interesting example since the GA has to 'discover' that 3 of the 5 variables are irrelevant. Considering the composition of the final population the Kriging functions

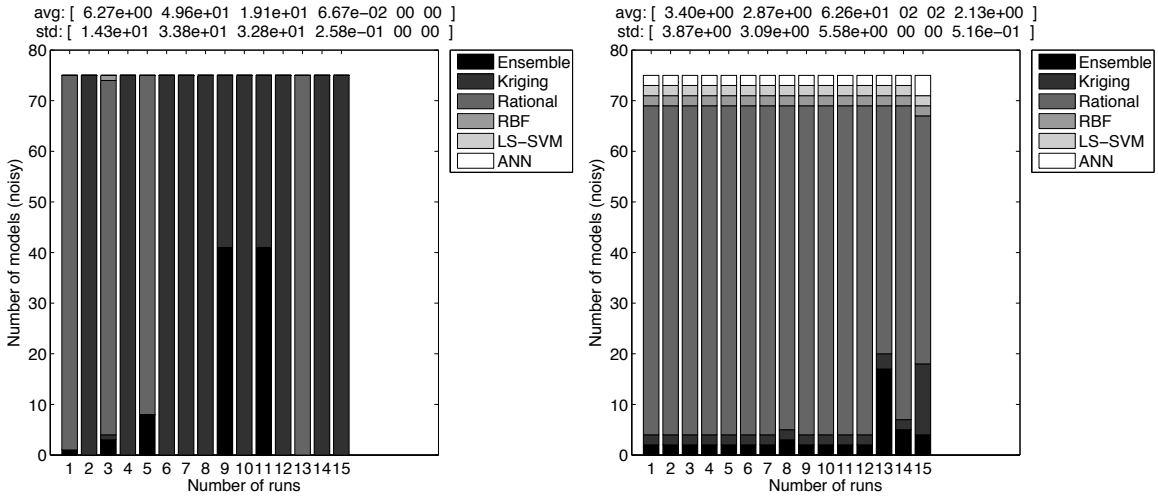


Figure 10: KF: Composition of the final population (Left: *EP=false*, Right: *EP=true*)

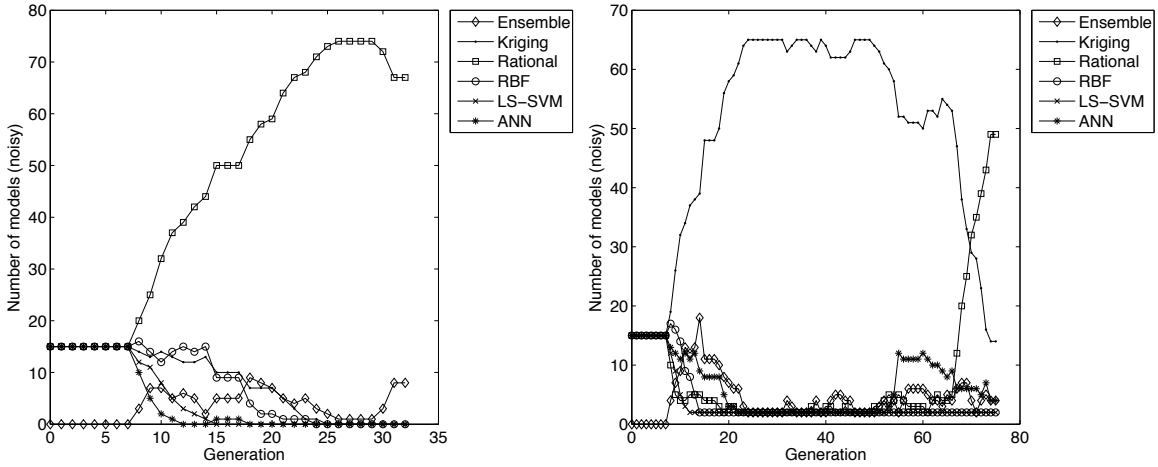


Figure 11: KF: Population evolution of the best run (Left: *EP=false*, Right: *EP=true*)

seem to be able to do this best in the *EP=false* case, with sporadic 'wins' for rational functions. In the *EP=true* case the situation is different, rational functions dominating all 15 runs. The fact that the rational functions succeed in doing this is thanks to a weighting scheme used in the genetic operators and described further in Hendrickx et al. (2006).

The usefulness of EP is demonstrated again as well. While the results of the best run for *EP=false* are better than the best run for *EP=true* (less samples), the former is much more a product of chance than the latter (which has lower variance). *EP=true* should still be preferred as it is more robust. Finally, the quality of the final models is excellent in all runs, and the performance and running time of the HGA remains competitive with the single model type runs.

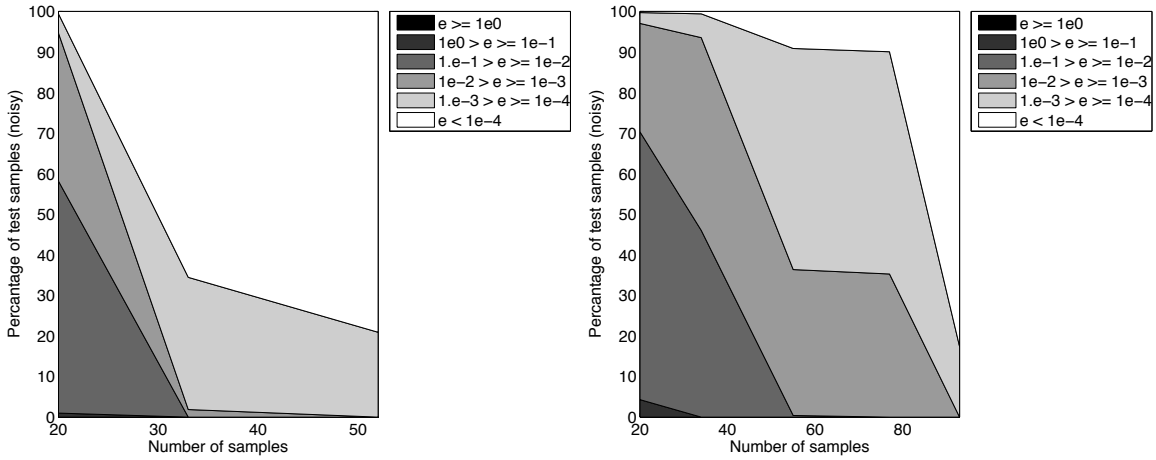


Figure 12: KF: Error evolution of the best run (Left: *EP=false*, Right: *EP=true*)

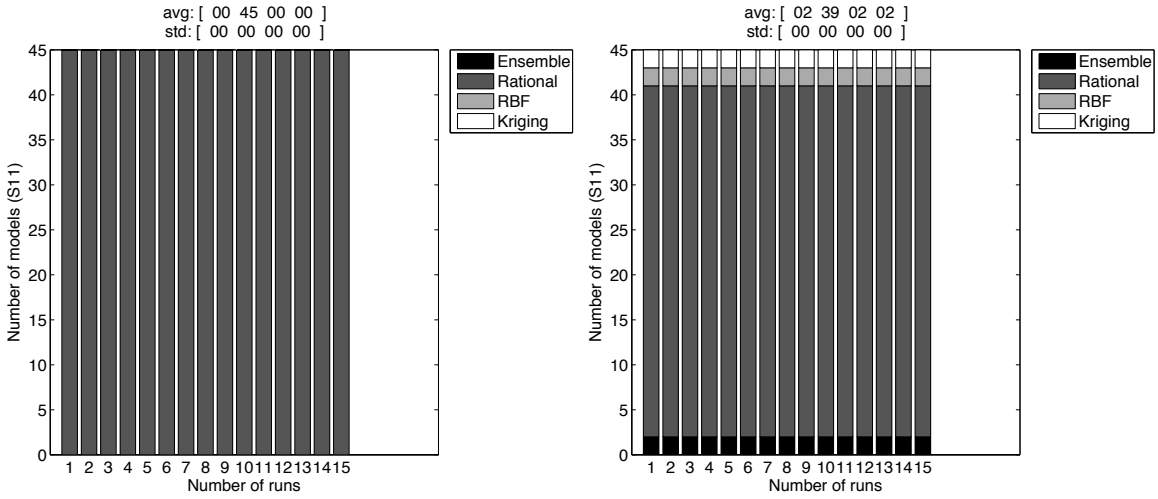


Figure 13: EE: Composition of the final population (Left: *EP=false*, Right: *EP=true*)

10.3 EM Example

The composition of the final population for each run is shown in Figure 13, and the associated error histogram in Figure 14. The population evolution and corresponding error evolution for the best run are shown in Figures 15 and 16. Table 3 summarizes the results and a plot of the best model can be found in Figure 17. Note that Table 3 shows the cross validation error (*CV*) instead of the validation error.

The results are very clear cut, rational functions dominate in every run, easily reaching the accuracy requirements in about 200 data points (with the *EP=true* runs generally reaching higher accuracies). This is to be expected. The physical behavior of two inductive posts in a rectangular waveguide is well described by a quotient of two differential multinomials (= the transfer function) and it is this function that needs to be modeled. Thus it is not surprising that rational functions do well since their form fits the underlying function.

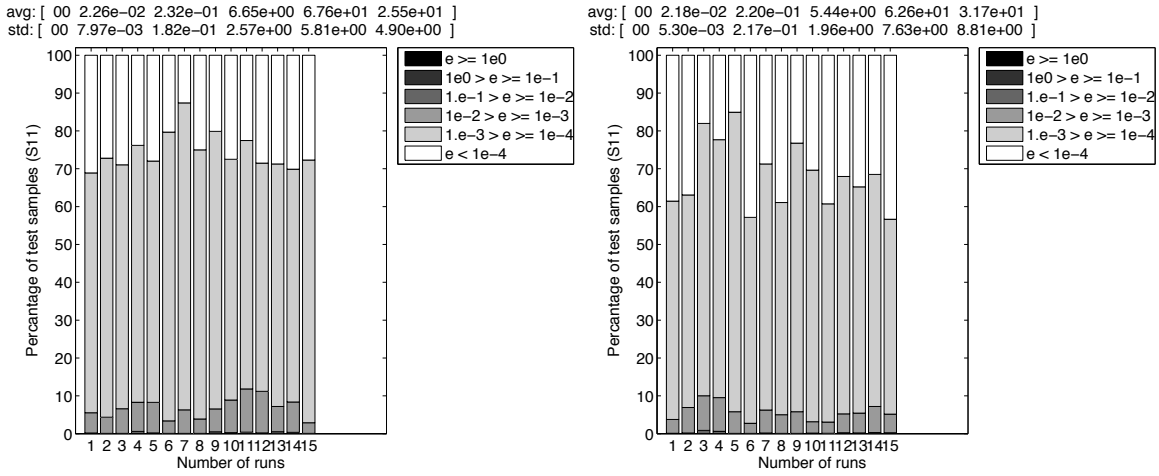


Figure 14: EE: Error histogram of the final best model in each run (Left: *EP=false*, Right: *EP=true*)

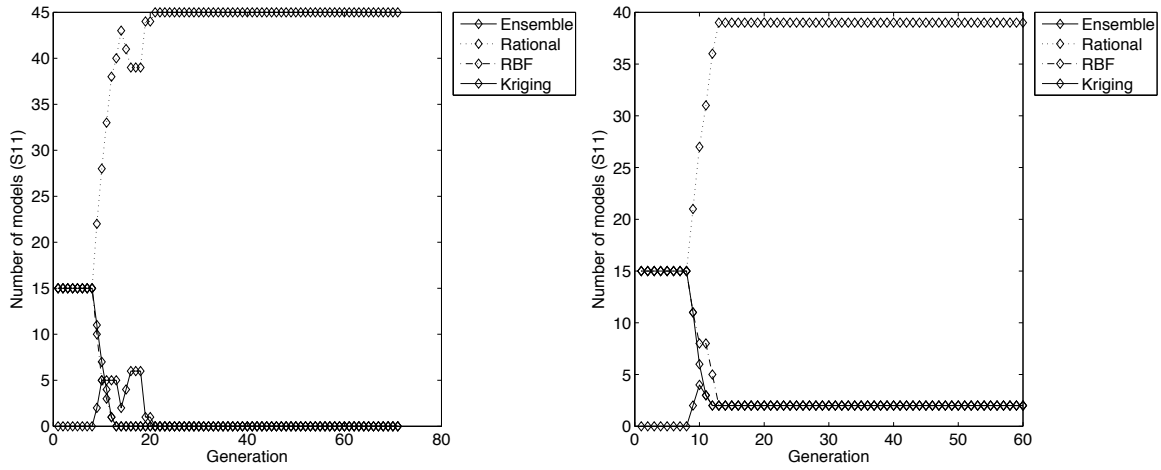


Figure 15: EE: Population evolution of the best run (Left: *EP=false*, Right: *EP=true*)

If we compare the HGA runs with the single model type runs we see significant improvements. Interestingly, the HGA runs need roughly 33-25% less sample evaluations to reach the target accuracy, and do so in a fraction of the time (less then 8 minutes vs. an average of 43 minutes for the homogeneous runs). Thus here we have a strong case for the use of the HGA.

10.4 LGBB Example

The composition of the final population for each run is shown in Figure 18 and the associated error histogram in Figure 19. The population evolution of the best run is shown in Figure 20. Table 4 shows the comparison with the homogeneous runs. A plot of the response can be found in Figure 21.

Adaptive sampling was switched off for the LGBB example. The objective was to see what accuracy can be reached and what model type prevails within a fixed time budget. The LGBB

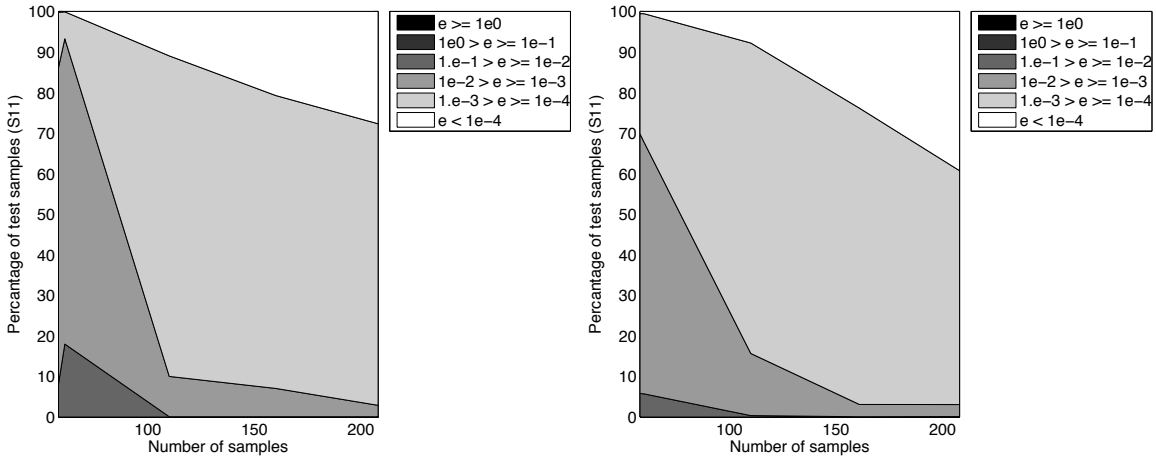


Figure 16: EE: Error evolution of the best run (Left: *EP=false*, Right: *EP=true*)

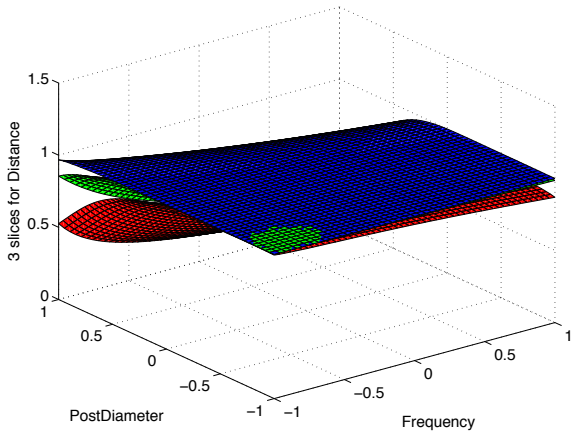


Figure 17: EE: normalized plot of $|S_{11}|$ of the best model overall ($HGA_{EP=true}$, 3 slices for *Distance*)

Method	$\|X\|$	σ	$\bar{CV}_{RRSE}$	σ	$\bar{TE}_{RRSE}$	σ	$\bar{time}$ *(min)*	σ
Kriging	7.980E+02	1.137E+02	8.541E-03	7.926E-04	1.881E-02	2.833E-03	7.202E+01	2.780E+01
Rational	8.147E+02	2.314E+02	1.152E-02	4.128E-03	1.708E-02	4.976E-03	4.046E+01	1.914E+01
RBF	6.080E+02	4.226E+01	8.123E-03	6.333E-04	1.556E-02	2.403E-03	1.713E+01	2.159E+00
$HGA_{EP=false}$	1.880E+02	2.536E+01	6.297E-03	1.770E-03	3.518E-02	4.571E-02	7.907E+00	1.440E+00
$HGA_{EP=true}$	1.980E+02	2.070E+01	6.733E-03	1.457E-03	2.227E-02	1.149E-02	7.722E+00	1.079E+00

Table 3: EE: Comparison with homogeneous evolution

example consists of a 3 dimensional data set and unlike the AF and EE examples there are no clues as to which model type is most adequate. Running the heterogeneous evolutionary algorithm it turns out that ANNs give the best fit overall (see Figure 18), achieving excellent accuracy. Changing the *EP* setting does not influence this, though the variance is lower for the *EP=true* case.

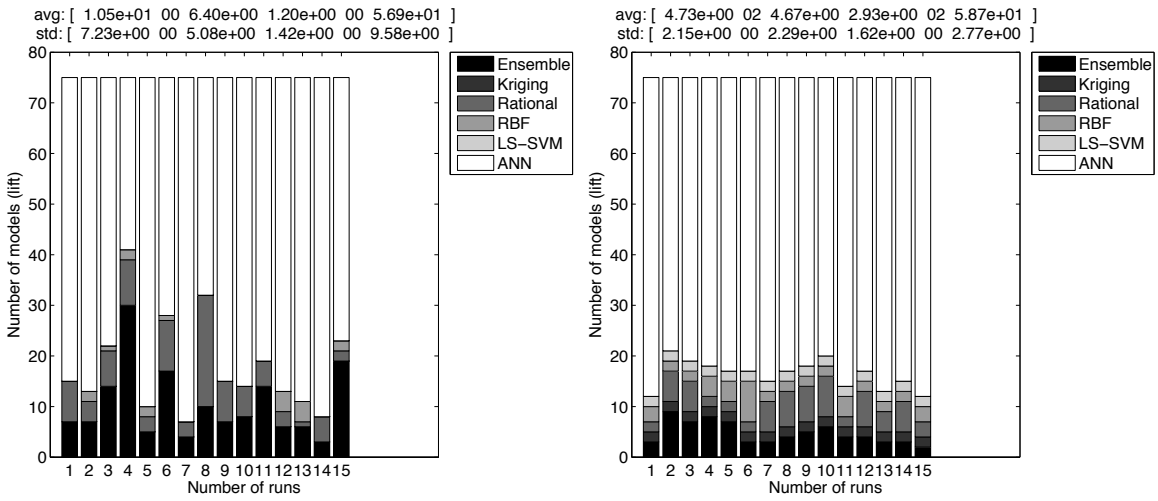


Figure 18: LE: Composition of the final population (Left: *EP=false*, Right: *EP=true*)

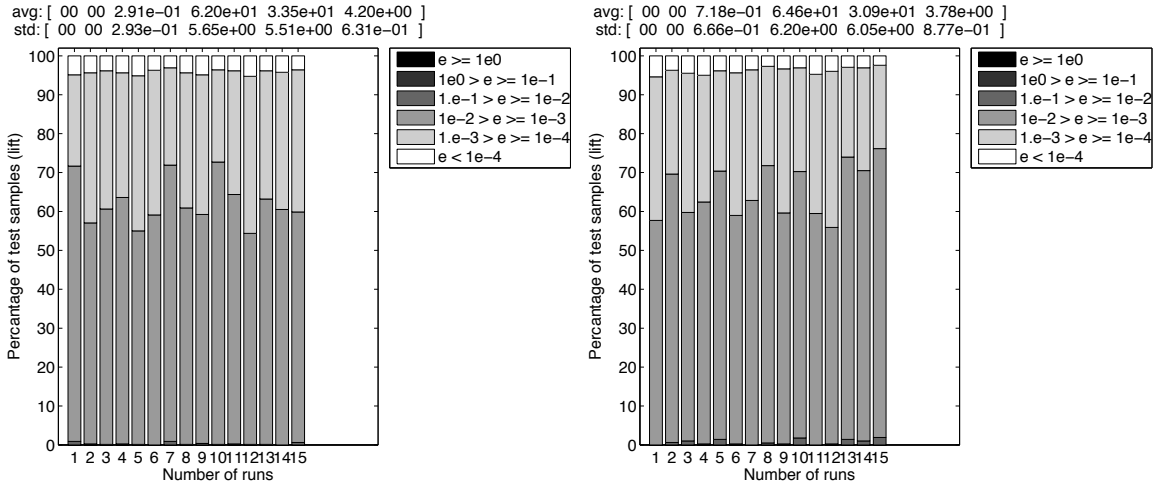


Figure 19: LE: Error histogram of the final best model in each run (Left: *EP=false*, Right: *EP=true*)

Method	$\|X\|$	$\bar{VE}_{RRSE}$	σ	$\bar{time}$ *(min)*
ANN	7.800E+02	7.47E-003	5.60E-004	3.00E+002
Kriging	7.800E+02	1.08E-001	1.96E-002	3.00E+002
LS-SVM	7.800E+02	1.40E-001	4.09E-005	3.00E+002
Rational	7.800E+02	5.20E-002	3.02E-004	3.00E+002
RBF	7.800E+02	7.34E-002	3.53E-008	3.00E+002
$HGA_{EP=false}$	7.800E+02	7.68E-003	4.38E-004	3.00E+002
$HGA_{EP=true}$	7.800E+02	7.59E-003	4.26E-004	3.00E+002

Table 4: LE: Comparison with homogeneous evolution

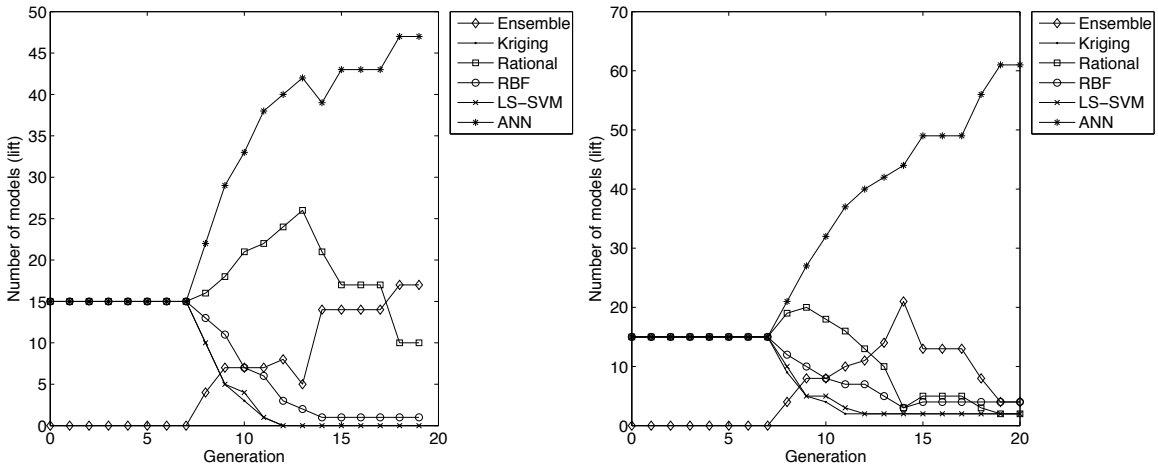


Figure 20: LE: Population evolution of the best run (Left: *EP=false*, Right: *EP=true*)

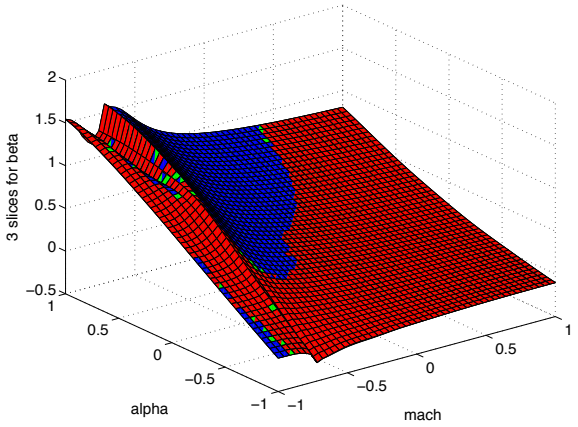


Figure 21: LE: normalized lot of the best model overall ($HGA_{EP=true}$, 3 slices for *beta*)

Within the same time limits the models produced by the HGA are comparable in accuracy to the best performing homogeneous runs, which again demonstrates the usefulness of the HGA.

Interestingly it turns out that the third dimension is negligible, the three slices in Figure 21 almost coincide. This was confirmed by using the SUMO model browser to fully explore the response. Thus we can safely conclude that side-slip angle has little or no effect on the lift on re-entry of the LGBB into the atmosphere.

10.5 Boston Housing Example

The final example is the Boston Housing data set, adaptive sampling was also switched off. The composition of the final population for each run is shown in Figure 22 and the associated error histogram in Figure 23. The population evolution of the best run is shown in Figure 24. Table 5 shows the comparison with the homogeneous runs.

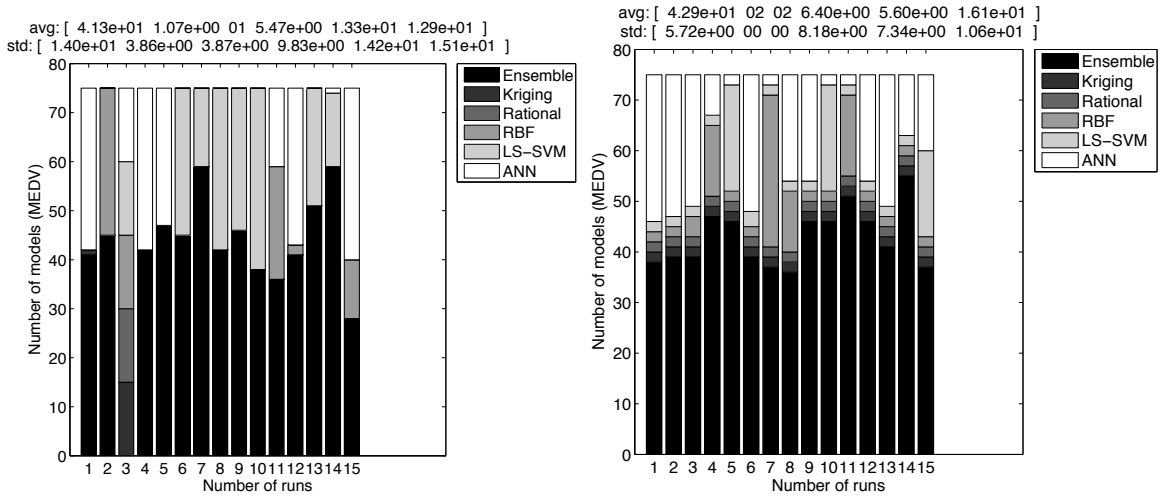


Figure 22: BH: Composition of the final population (Left: *EP=false*, Right: *EP=true*)

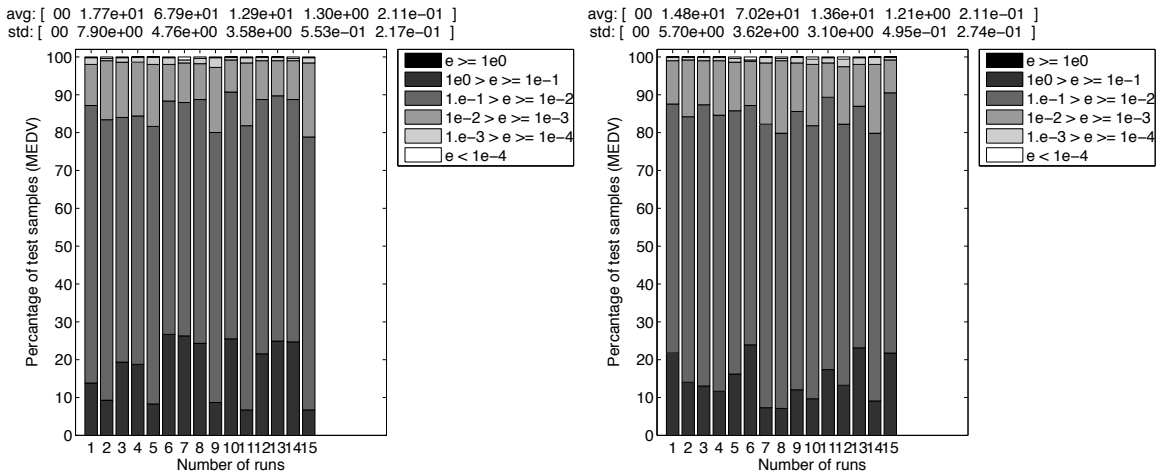


Figure 23: BH: Error histogram of the final best model in each run (Left: *EP=false*, Right: *EP=true*)

Method	$\|X\|$	$\bar{VE}_{RRSE}$	σ	$\bar{time}$ *(min)*
ANN	5.060E+02	2.985E-01	8.962E-03	1.200E+03
Kriging	5.060E+02	3.448E-01	1.077E-02	1.200E+03
LS-SVM	5.060E+02	3.421E-01	6.012E-06	1.200E+03
Rational	5.060E+02	5.006E-01	4.296E-02	1.200E+03
RBF	5.060E+02	1.228E+15	2.104E+15	1.200E+03
$HGA_{EP=false}$	5.060E+02	2.764E-01	2.768E-02	1.200E+03
$HGA_{EP=true}$	5.060E+02	2.735E-01	1.372E-02	1.200E+03

Table 5: BH: Comparison with homogeneous evolution

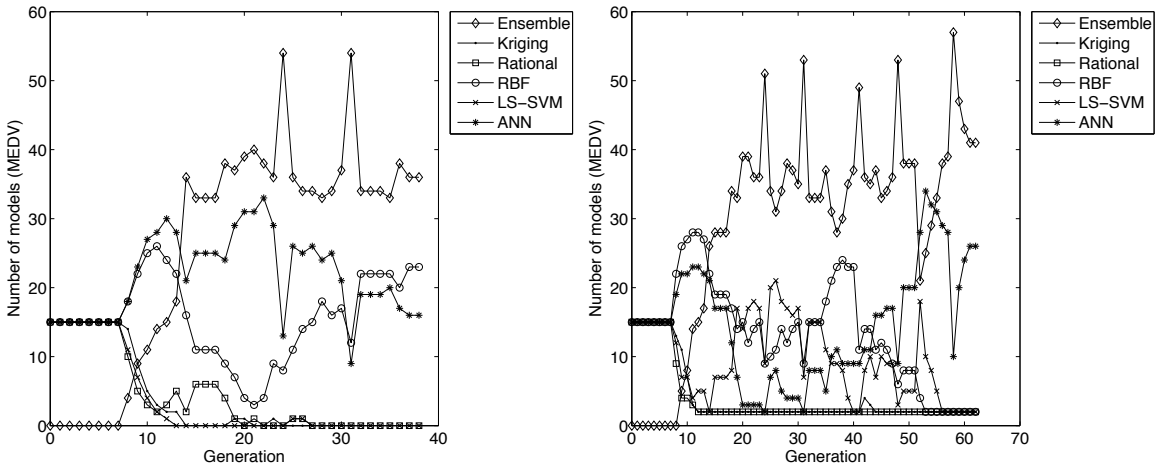


Figure 24: BH: Population evolution of the best run (Left: *EP=false*, Right: *EP=true*)

This is a somewhat curious example since it has high dimensionality (13), small support (506 tuples), and the types and ranges of the different inputs parameters vary greatly (e.g., input 4 (CHAS) is a boolean variable that is 1 if the tract borders the river and 0 otherwise while input 5 (NOX) is the nitric oxide concentration). Consequently, any analysis of this data should be preceded by a thorough statistical treatment (feature selection, variance analysis, etc.). We explicitly chose not to do this but take the data as is and treat is as black box regression problem.

The results are mixed (*see* Figure 22), though the HGA runs again outperform the homogeneous runs. (LS-)SVM and ANN models seem to be preferred over Kriging and RBF models but there is no evidence to distinguish between the models any further. Striking, though, is that about half of the final population consists of ensembles and that most of these ensembles turn out to be {ANN, RBF} pairs or multiple ANNs. Figure 25 shows the evolution of the composition of the best performing ensemble. The popularity of ensembles in this case is in line with the authors' previous experiences. When the individual model types are having trouble to fit a difficult response with none really performing much better than the other, hybrids (ensembles) tend to do well since they can produce more complicated responses. It is a signal that none of the included model types are really fit for the approximation problem.

Also striking (and interesting) are the oscillations in the population evolution (*see* Figure 24, or Figure 26 for a more marked example). It turns out that every run shows these oscillations between ensembles and one or two other model types. Interestingly these 'spikes' occur every 10 or 7 generations. It remains unclear to the authors how these oscillations may be explained. This is an issue that is being investigated in more detail.

11. Summary

In summary the results for the different test problems are very promising and in line with previous results (Gorissen, 2007; Gorissen et al., 2008a; Couckuyt et al., 2009; Gorissen et al., 2009a). The results show a consensus about which model type to use in all test cases (ignoring the BH example for the moment). In the case the consensus is not absolute (e.g., the *EP=true* run for the AF in Figure 4) the final model accuracies are essentially the same thus this is not really a problem from

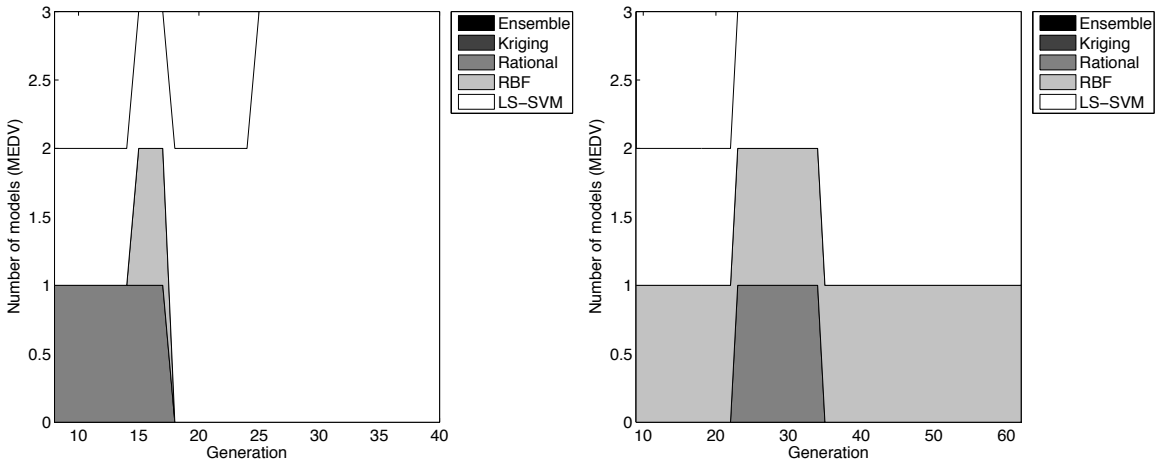


Figure 25: BH: Composition of the best performing ensemble of the best run (Left: *EP=false*, Right: *EP=true*)

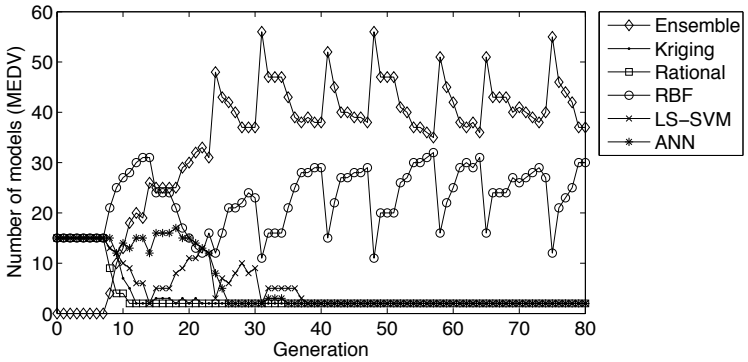


Figure 26: BH: Oscillations in the population evolution ($HGA_{EP=true}$)

an application standpoint. More important is that the target accuracy has been reached and that all model types have been given a fair chance without having to resort to a brute force approach.

In general we found the algorithm to be quite consistent across many runs. When variation does show up in the model selection results it typically is because two or more model types can fit the data equally well with only a minor difference in accuracy. This means that the GA may alternate between the different local optima, giving different model selection results, but still reaching the targets. The other reason is if the data is simply too difficult to fit using the methods included in the evolution. In this case ensembles may tend to do well. The BH example seems indicative of this situation.

It is important to remind the reader, though, that the overall performance of the HGA will of course depend on the quality of the model types themselves, and more importantly, on the quality of the creation function and genetic operators (and adequacy of the chosen representation). Good results have already been obtained with the current implementations though there is still room for

improvement with respect to the application by an expert in any one type.[5] Luckily, an advantage of the HGA based approach is that since the general algorithm is now in place, it becomes possible to focus on such improvements without requiring changes to the HGA itself. Specific improvements (e.g., devised by an expert in a certain model type) are straightforward to integrate into the existing genetic operators allowing an accumulation knowledge that will improve the overall quality of the models produced by the HGA.

A next step is to further increase the number of test problems and, more importantly, investigate the influence of the different parameters involved. In this respect the migration interval and migration topology parameters are particularly important since they determine how the different model types interact. For example, if the migration interval is set too high, each deme will produce high quality models (most of the time is spent optimizing the parameters of a single model type) but there will have been very little competition between models. If it is set too low, the converse is true. More research is needed to to better understand this balance and investigate the impact of genetic drift.

12. Conclusion and Future Work

Due to the computational complexity of current simulation codes, the use of global surrogate modeling techniques (adaptive sampling, adaptive modeling) has become standard practice among scientists and engineers alike. However, a recurring problem is selecting the most adequate surrogate model type and associated complexity. In this contribution we explored an approach based on the evolutionary migration model that can help tackle this problem in an automatic way if little information is known about the true response behavior and there are no a priori model type requirements. In addition, we have illustrated the usefulness of extinction prevention and ensemble based recombination. Extinction prevention is a straightforward algorithm that prevents a species from disappearing from the gene pool at the expense of a minor cost (keeping 2 extra individuals per species 'alive'). As a result, the optimal solution is able to change with time, making for a more flexible and adaptive system which, as demonstrated in the different examples, gives better and more consistent results.

Future work will consist of investigating the oscillations in the BH example, exploring different GA parameter values (role of the migration frequency, migration topology, etc.), incorporating more model types, and more advanced ensemble methods (e.g., stronger constraints on ensemble composition). As mentioned above, improvements to the genetic operators are ongoing in order to get more out of each model type. The utility of adding a penalty to the fitness function proportional to the model complexity and/or training time will also be investigated. Furthermore, we have been experimenting with sampling strategies that vary dynamically depending on the remaining sample budget and quality & type of surrogate currently used in the modeling process. The idea is to work towards an optimal interplay between sampling and modeling. E.g., initially the focus should be on exploration of the design space while, as accuracy of the models improves, the focus should shift towards refining the model in places where it is uncertain and ensuring the optima it exhibits are really true optima. Likewise, we are experimenting with dynamic model selection criteria. For example, if only little data is available cross validation type measures may be unreliable and it makes little sense enforcing problem specific constraints (e.g., the model response should be bounded between given bounds). However, when the data density is sufficiently high the opposite will be true. Thus there seems to be some intuition advocating the use of annealing type strategies.

5. There is a trade-off involved here. Expert application of a surrogate model type will invariably lead to problem specific bias, reducing the performance on other problems.

In addition to the global modeling use case. We are also experimenting with linking the HGA described here with the EGO framework from Jones et al. (1998). The structure of the SUMO Toolbox allows natural linking of these two components. This allows for automatic model type switching during optimization (any model type that supports prediction variance can be used) and may be beneficial for computational expensive codes.

Finally it should be noted that all the algorithms and examples described here are available for download at `http://www.sumo.intec.ugent.be`.

Acknowledgments

The authors would like to thank NASA and Robert Gramacy from the University of California, Santa Cruz for providing the data on the LGBB example. Special thanks also to Robert Lehmensiek and Petrie Meyer from the University of Stellenbosch, South Africa, for providing the simulation code for the EM example.

This work was supported by the Fund for Scientific Research Flanders (FWO Vlaanderen) and the Science and Innovation Administration Flanders (BSTC project).

References

E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 6(5):443–462, 2002.

C. A. Ankenbrandt. An extension to the theory of convergence and a proof of the time complexity of genetic algorithms. In *Foundations of Genetic Algorithms*, pages 53–68, 1990.

Y. B., R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004.

D. Busby, C. L. Farmer, and A. Iske. Hierarchical nonlinear approximation for experimental design and statistical data fitting. *SIAM Journal on Scientific Computing*, 29(1):49–69, 2007. doi: 10.1137/050639983.

P-W. Chen, J-Y. Wang, and H-M. Lee. Model selection of SVMs using GA approach. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 3, pages 2035–2040, 25-29 July 2004.

V. Chen, K-L. Tsui, R. Barton, and M. Meckesheimer. A review on design, modeling and applications of computer experiments. *IIE Transactions*, 38:273–291, 2006.

H-S. Chung and J. J. Alonso. Comparison of approximation models with merit functions for design optimization. In *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, September 2000. AIAA Paper 2000-4754.

S. Conti and A. O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. research report no. 569/07, submitted to Journal of Statistical Planning and Inference. Technical report, Department of Probability and Statistics, University of Sheffield, 2007.

I. Couckuyt, D. Gorissen, H. Rouhani, E. Laermans, and T. Dhaene. Evolutionary regression modeling with active learning: An application to rainfall runoff modeling. In *International Conference on Adaptive and Natural Computing Algorithms*, volume LNCS 5495, pages 548–558, 2009.

K. Crombecq, D. Gorissen, L. De Tommasi, and T. Dhaene. A novel sequential design strategy for global surrogate modeling. In *Proceedings of the 41th Conference on Winter Simulation, accepted*, 2009.

D. Deschrijver and T. Dhaene. Rational modeling of spectral data using orthonormal vector fitting. In *Signal Propagation on Interconnects, 2005. Proceedings. 9th IEEE Workshop on*, pages 111–114, 10-13 May 2005. doi: 10.1109/SPI.2005.1500915.

M. S. Eldred and D. M. Dunlavy. Formulations for surrogate-based optimization wiht data fit, multifidelity, and reduced-order models. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Protsmouth, Virginia*, 2006.

M. T. M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Trans. Evolutionary Computation*, 10(4):421–439, 2006.

H. J. Escalante, M. M. Gomez, and L. E. Sucar. PSMS for neural networks. In *The IJCNN 2007 Agnostic vs Prior Knowledge Challenge*, pages 678–683, 2007.

H. Jair Escalante, M. Montes, and E. Sucar. Particle swarm model selection. *Journal of Machine Learning Research (special issue on model selection)*, 10:405–440, 2008.

H. Fang, M. Rais-Rohani, Z. Liu, and M.F. Horstemeyer. A comparative study of metamodeling methods for multiobjective crashworthiness optimization. *Computers and Structures*, 83:2121–2136, 2005.

F.D. Foresee and M.T. Hagan. Gauss-newton approximation to bayesian regularization. In *Proceedings of the 1997 International Joint Conference on Neural Networks*, pages 1930–1935, 1997.

A. Forrester, A. Sobester, and A. Keane. *Engineering Design Via Surrogate Modelling: A Practical Guide*. Wiley, 2008.

F. Friedrichs and C. Igel. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 64: 107–117, 2005.

S.E. Gano, H. Kim, and D.E. Brown. Comparison of three surrogate modeling techniques: Datascape, kriging, and second order regression. In *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA-2006-7048, Portsmouth, Virginia*, 2006.

M. Ganser, K. Grossenbacher, M. Schutz, L. Willmes, and T. Back. Simulation meta-models in the early phases of the product development process. In *Proceedings of Efficient Methods for Robust Design and Optimization (EUROMECH 07)*, 2007.

K. C. Giannakoglou, M. K. Karakasis, and I. C. Kampolis. Evolutionary algorithms with surrogate modeling for computationally expensive optimization problem. In *Proceedings of ERCOFTAC 2006 Design Optimization International Conference, Gran Canaria, Spain*, 2006.

T. Goel, R. Haftka, W. Shyy, and N. Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33:199–216, 2007. doi: doi:10.1007/s00158-006-0051-9.

D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989. ISBN 0201157675.

D. Gorissen. Heterogeneous evolution of surrogate models. Master's thesis, Master of AI, Katholieke Universiteit Leuven (KUL), 2007.

D. Gorissen, L. De Tommasi, J. Croon, and T. Dhaene. Automatic model type selection with heterogeneous evolution: An application to RF circuit block modeling. In *Proceedings of the IEEE Congress on Evolutionary Computation, WCCI 2008, Hong Kong*, 2008a.

D. Gorissen, L. De Tommasi, W. Hendrickx, J. Croon, and T. Dhaene. RF circuit block modeling via kriging surrogates. In *Proceedings of the 17th International Conference on Microwaves, Radar and Wireless Communications (MIKON 2008)*, 2008b.

D. Gorissen, I. Couckuyt, K. Crombecq, and T. Dhaene. Pareto-based multi-output model type selection. In *Proceedings of the 4th International Conference on Hybrid Artificial Intelligence (HAIS 2009), Salamanca, Spain*, pages 442–449. Springer - Lecture Notes in Artificial Intelligence, Vol. LNCS 5572, 2009a.

D. Gorissen, I. Couckuyt, E. Laermans, and T. Dhaene. Multiobjective global surrogate modeling,dealing with the 5-percent problem - in press. *Engineering with Computers*, 2009b.

D. Gorissen, L. De Tommasi, K. Crombecq, and T. Dhaene. Sequential modeling of a low noise amplifier with neural networks and active learning. *Neural Computing and Applications*, 18(5): 485–494, 2009c.

R. Gramacy, H. Lee, and W. Macready. Parameter space exploration with gaussian process trees. In *ICML '04: Proceedings of the 21st International Conference on Machine Learning*, page 45, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-828-5. doi: http://doi.acm.org/10.1145/1015330.1015367.

L. Gu. A comparison of polynomial based regression models in vehicle safety analysis. In A. Diaz, editor, *2001 ASME Design Automation Conference, ASME, Pittsburgh, PA*, 2001.

D. Harrison and D.L. Rubinfeld. Hedonic prices and the demand for clean air. *Journal of Environmental Economics & Management*, 5:81–102, 1978.

W. Hendrickx, D. Gorissen, and T. Dhaene. Grid enabled sequential design and adaptive metamodeling. In *WSC '06: Proceedings of the 37th Conference on Winter simulation*, pages 872–881. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7.

C. Hocaoglu and A. Sanderson. Planning multiple paths with evolutionary speciation. *IEEE Trans. Evolutionary Computation*, 5(3):169–191, 2001.

J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

R. Jin, W. Chen, and T.W. Simpson. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization*, 23(1):1–13, December 2001. doi: 10.1007/s00158-001-0160-4.

Yaochu Jin and B. Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(3):397–415, May 2008. ISSN 1094-6977. doi: 10.1109/TSMCC.2008.919172.

Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Trans. Evolutionary Computation*, 6(5):481–494, 2002.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998. ISSN 0925-5001. doi: http://dx.doi.org/10.1023/A:1008306431147.

A. C. Keys, L. P. Rees, and A. G. Greenwood. Performance measures for selection of metamodels to be used in simulation optimization. *Decision Sciences*, 33:31 – 58, 2007.

J. Kleijnen, S. Sanchez, T. Lucas, and T. Cioppa. State-of-the-art review: A user's guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing*, 17(3): 263–289, 2005.

J. Knowles and H. Nakayama. Meta-modeling in multiobjective optimization. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 245–284. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-88907-6. doi: http://dx.doi.org/10.1007/978-3-540-88908-3_10.

R. Lehmensiek. *Efficient Adaptive Sampling Applied to Multivariate, Multiple Output Rational Interpolation Models, with Applications in Electromagnetics-based Device Modeling*. PhD thesis, University of Stellenbosch, 2001.

S. Lessmann, R. Stahlbock, and S.F. Crone. Genetic algorithms for support vector machine model selection. In *Proceedings of the International Joint Conference on Neural Networks, 2006. IJCNN '06.*, pages 3063–3069, 16-21 July 2006.

X. Rong Li and Z. Zhao. Evaluation of estimation algorithms part I: incomprehensive measures of performance. *IEEE Transactions on Aerospace and Electronic Systems*, 42(4):1340–1358, October 2006. ISSN 0018-9251. doi: 10.1109/TAES.2006.314576.

D. Lim, Y-S. Ong, Y. Jin, and B. Sendhoff. A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 07)*, pages 1288–1295, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-697-4. doi: http://doi.acm.org/10.1145/1276958.1277203.

S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Aspects of the matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002.

D. MacKay. Bayesian model comparison and backprop nets. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 839–846, San Mateo, California, 1992. Morgan Kaufmann.

T. Nakama. Theoretical analysis of genetic algorithms in noisy environments based on a markov model. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO 08)*, pages 1001–1008, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-130-9. doi: http://doi.acm.org/10.1145/1389095.1389283.

A. Neubauer. A theoretical analysis of the non-uniform mutation operator for the modified genetic algorithm. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 93–96, Apr 1997. doi: 10.1109/ICEC.1997.592275.

M. Nowostawski and R. Poli. Parallel genetic algorithm taxonomy. In *Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference*, pages 88–92, 31 Aug.-1 Sept. 1999. doi: 10.1109/KES.1999.820127.

Y-S. Ong, P.B. Nair, and K.Y. Lum. Max-min surrogate-assisted evolutionary algorithm for robust design. *Evolutionary Computation, IEEE Transactions on*, 10(4):392–404, Aug. 2006. doi: 10.1109/TEVC.2005.859464.

I. Paenke, J. Branke, and Y. Jin. Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. *IEEE Trans. Evolutionary Computation*, 10(4):405–420, 2006.

B. Pamadi, P. Covell, P. Tartabini, and K. Murphy. Aerodynamic characteristics and glide-back performance of langley glide-back booster. In *Proceedings of 22nd Applied Aerodynamics Conference and Exhibit, Providence, Rhode Island*, 2004.

H. Pei and E. Goodman. A comparison of cohort genetic algorithms with canonical serial and island-model distributed ga's. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 501–510, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann. ISBN 1-55860-774-9.

R. Poli. Exact schema theory for genetic programming and variable-length genetic algorithms with one-point crossover. *Genetic Programming and Evolvable Machines*, 2(2):123–163, 2001. ISSN 1389-2576. doi: http://dx.doi.org/10.1023/A:1011552313821.

X. Qi and F. Palmieri. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space. part I: Basic properties of selection and mutation. *Neural Networks, IEEE Transactions on*, 5(1):102–119, Jan 1994a. ISSN 1045-9227. doi: 10.1109/72.265965.

X. Qi and F. Palmieri. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space. part II: Analysis of the diversification role of crossover. *Neural Networks, IEEE Transactions on*, 5(1):120–129, Jan 1994b. ISSN 1045-9227. doi: 10.1109/72.265966.

N. Queipo, R. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Tucker. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41:1–28, 2005.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

G. Rawlins, editor. *Foundations of Genetic Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991. ISBN 1558605592.

R.G. Regis and C.A. Shoemaker. Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Trans. Evolutionary Computation*, 8(5):490–505, 2004.

S. Rogers, M. Aftosmis, S. Pandya, and N. Chaderjian. Automated CFD parameter studies on distributed parallel computers. In *Proc of.16th AIAA Computational Fluid Dynamics Conference, Orlando, Florida*, 2003.

E. Sanchez, S. Pintos, and N.V. Queipo. Toward an optimal ensemble of kernel-based approximations with engineering applications. In *In Proceedings of the International Joint Conference on Neural Networks, 2006. IJCNN '06.*, pages 2152–2158, 2006. doi: 10.1109/IJCNN.2006.246987.

T. Santner, B. Williams, and W. Notz. *The design and analysis of computer experiments*. Springer series in statistics. Springer, 2003.

A. Sharkey. On combining artificial neural nets. *Connectionist Science*, 8(3):299–314, 1996.

T. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen. Metamodels for computer-based engineering design: Survey and recommendations. *Eng. Comput. (Lond.)*, 17(2):129–150, 2001.

T. W. Simpson, V. Toropov, V. Balabanov, and F. A. C. Viana. Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come or not. In *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2008 MAO, Victoria, Canada*, 2008.

G. Smits and M. Kotanchek. Pareto-front exploitation in symbolic regression. In *Genetic Programming Theory and Practice II*. Springer, Ann Arbor, USA, 2004.

D. P. Solomatine and A. Ostfeld. Data-driven modelling : some past experiences and new approaches. *Journal of hydroinformatics*, 10(1):3–22, 2008.

K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002. ISSN 1063-6560. doi: http://dx.doi.org/10.1162/106365602320169811.

M. Streeter and L. Becker. Automated discovery of numerical approximation formulae via genetic programming. *Genetic Programming and Evolvable Machines*, 4(3):255–286, 2003. ISSN 1389-2576. doi: http://dx.doi.org/10.1023/A:1025176407779.

J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific Publishing Co., Pte, Ltd., Singapore, 2002. ISBN 981-238-151-1.

D. J.J. Toal, N. W. Bressloff, and A. J. Keane. Kriging hyperparameter tuning strategies. *AIAA Journal*, 46(5):1240–1252, 2008.

S. Tomioka, S. Nisiyama, and T. Enoto. Nonlinear least square regression by adaptive domain method with multiple genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 11 (1):1–16, February 2007.

P. Triverio, S. Grivet-Talocia, M.S. Nakhla, F. G. Canavero, and R. Achar. Stability, causality, and passivity in electrical interconnect models. *IEEE Transactions on Advanced Packaging*, 30(4): 795–808, 2007.

C. Turner, R. Crawford, and M. Campbell. Multidimensional sequential sampling for nurbs-based metamodel development. *Engineering with Computers*, 23(3):155–174, 2007. ISSN 0177-0667. doi: http://dx.doi.org/10.1007/s00366-006-0051-9.

E.J. Vladislavleva, G.F. Smits, and D. den Hertog. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *Evolutionary Computation, IEEE Transactions on*, 13(2):333–349, April 2009. ISSN 1089-778X. doi: 10.1109/TEVC.2008.926486.

I. Voutchkov and A.J. Keane. Multiobjective Optimization using Surrogates. In I.C. Parmee, editor, *Adaptive Computing in Design and Manufacture 2006. Proceedings of the Seventh International Conference*, pages 167–175, Bristol, UK, April 2006.

G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007. doi: 10.1115/1.2429697.

D. Whitley, S. Rana, and R. B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7(1):33–47, 1999.

Y. Xiong, W . Chen, D. Apley, and X. Ding. A nonstationary covariance-based kriging method for metamodeling in engineering design. *International Journal for Numerical Methods in Engineering*, 71(6):733–756, 2007.

R.J. Yang, N. Wang, C. H. Tho, and J. P. Bobinaeu. Metamodeling development for vehicle frontal impact simulation. *Journal of Mechanical Design*, 127(5):1014–1020, September 2005.

X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, Sept. 1999. doi: 10.1109/5.784219.

X. Yao and Y. Xu. Recent advances in evolutionary computation. *Journal of Computer Science and Technology*, 21(1):1–18, 2006.

K. Ye, W. Li, and A. Sudjianto. Algorithmic construction of optimal symmetric latin hypercube designs. *Journal of Statistical Planning and Inference*, 90:145–159, 2000.

Y-S. Yeun, W-S. Ruy, Y-S. Yang, and N-J. Kim. Implementing linear models in genetic programming. *IEEE Trans. Evolutionary Computation*, 8(6):542–566, 2004.

C. Zhang, H. Shao, and Y. Li. Particle swarm optimisation for evolving artificial neural network. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 4, pages 2487–2490vol.4, 8-11 Oct. 2000. doi: 10.1109/ICSMC.2000.884366.

Journal of Machine Learning Research 10 (2009) 2079-2114 Submitted 6/08; Revised 4/09; Published 9/09

An Anticorrelation Kernel for Subsystem Training in Multiple Classifier Systems

Luciana Ferrer[*] LFERRER@SPEECH.SRI.COM
Speech Technology and Research Laboratory
SRI International
333 Ravenswood Ave
Menlo Park, California, 94025, USA

Kemal Sönmez SONMEZK@OHSU.EDU
Division of Biomedical Computer Science, School of Medicine
Oregon Health and Science University
3181 S.W. Sam Jackson Park Rd.
Portland, Oregon, 97239, USA

Elizabeth Shriberg EES@SPEECH.SRI.COM
Speech Technology and Research Laboratory
SRI International
333 Ravenswood Ave
Menlo Park, California, 94025, USA

Editor: Leon Bottou

Abstract

We present a method for training support vector machine (SVM)-based classification systems for combination with other classification systems designed for the same task. Ideally, a new system should be designed such that, when combined with existing systems, the resulting performance is optimized. We present a simple model for this problem and use the understanding gained from this analysis to propose a method to achieve better combination performance when training SVM systems. We include a regularization term in the SVM objective function that aims to reduce the average class-conditional covariance between the resulting scores and the scores produced by the existing systems, introducing a trade-off between such covariance and the system's individual performance. That is, the new system "takes one for the team", falling somewhat short of its best possible performance in order to increase the diversity of the ensemble. We report results on the NIST 2005 and 2006 speaker recognition evaluations (SREs) for a variety of subsystems. We show a gain of 19% on the equal error rate (EER) of a combination of four systems when applying the proposed method with respect to the performance obtained when the four systems are trained independently of each other.

Keywords: system combination, ensemble diversity, multiple classifier systems, support vector machines, speaker recognition, kernel methods

∗. This author performed part of the work presented in this paper while at the Information Systems Laboratory, Department of Electrical Engineering, Stanford University.

1. Introduction

The work presented in this paper is motivated by our work on the task of speaker verification. In the last decade, many successful speaker verification systems have relied on the combination of various component systems to achieve superior performance. In many cases, as in Ferrer et al. (2006) and Brummer et al. (2007), the combination leads to significant improvements. However, there are cases in which combining several comparably good systems does not result in improvements over the single best system (Reynolds et al., 2005). Most of these systems perform the combination of information sources at the score level[1] (Reynolds et al., 2003; Ferrer et al., 2006; Brummer et al., 2007; Huenupán et al., 2007; Dehak et al., 2007): systems that model each type of feature using a certain model are independently developed and their scores are combined to produce the final score and the decision. When training each individual system, all other systems available for combination are usually ignored while, in fact, the ultimate goal of the systems is to perform well in combination with all the other systems and not necessarily individually.

It is easy to see that system combination at the score level is not guaranteed to give strictly better performance than those of the individual systems being combined. In the extreme case, if all classifiers were generating exactly the same output for each sample, the combined classifier would not have better performance than the individual ones, independently of the combination procedure used. Intuitively, what we wish is to have enough diversity across systems such that classifiers contribute complementary information leading to a better final decision when systems are combined. System diversity has been the subject of a large amount of research in recent years, with two main goals: defining a measure of diversity that can predict the performance of the combination, and designing procedures for achieving diversity in an ensemble of systems. With the goal of motivating and placing our work in perspective, in the next section we present a discussion focused on existing techniques for measuring and designing for system diversity.

The contributions of this paper are: (1) the development of a simple model for the combination problem for a binary classification task under the assumption that the distribution of scores for each of the classes is Gaussian, and (2) a procedure for improving diversity in an ensemble including SVM classifiers. We find an upper bound on the EER of an ensemble combination and show that, for a two-system combination, this upper bound is a function of the performance of the individual systems and the correlation coefficient obtained from the average class-conditional covariance matrix of the scores from the two systems. Based on this result, we propose the inclusion of a regularization term in the SVM objective function when training a new SVM system for combination with a set of preexisting systems, which introduces a trade-off between the performance of the resulting model and its average class-conditional covariance with the preexisting systems. Ferrer et al. (2008b) presented empirical results using the proposed method. Here, we extend this previous work by developing a framework under which to understand the method, considering the cases of multiple preexisting systems and nonlinear kernels, and including new results on simulated and on the NIST 2005 and 2006 speaker-verification evaluation data. Results show that a gain of 19% on EER can be achieved when using the proposed method with respect to the results obtained when systems are trained without knowledge of the others.

The paper is organized as follows. Section 2 gives a review of the related research. Section 3 describes a simple model for the system combination problem under consideration. Using the

1. The term *score* is commonly used in the speaker verification community to refer to the numerical output of a system, which may or may not be a probability measure.

conclusions obtained from this development, Section 4 proposes a method for achieving improved combination performance. In Sections 5 and 6 we present results on simulated data and speaker verification data, respectively. Section 7 presents our conclusions.

2. Multiple Classifier Systems

In this section we review the literature related to our work, motivating and putting in perspective the research presented in the rest of the paper.

2.1 Measuring Diversity

For regression problems, the measures of ensemble diversity are well developed. Krogh and Vedelsby (1995) showed that the quadratic error for a certain input value of a convex combination of estimators trained on a single data set is guaranteed to be less than or equal to the weighted average quadratic error of the component estimators. The difference between the two is given by an ambiguity term that measures the variability among ensemble members for the particular input. On a related development, Ueda and Nakano (1996) give a decomposition of the mean square error of an ensemble classifier into three terms: average bias, variance and covariance of the ensemble members. One would then wish to reduce the covariance term without affecting the bias and variance terms by reducing the correlation between the members. This term can, in fact, be negative.

Classification problems where ensemble members output an estimate of the posterior probabilities of the classes (as opposed to just the estimated label) can also be considered regression problems. Tumer and Ghosh (1996) and, later, Fumera and Roli (2005) present a framework for studying this case. They consider the probability of error as the measure of performance. If the actual posterior probabilities of the classes given the features were known, the probability of error could be minimized by choosing, for each value of the input features, the class that has the highest posterior. This rule determines a boundary between the regions where each class is predicted and results in a certain probability of error that is called the *Bayes error*. In practice, the posterior probabilities are not known. Tumer and Ghosh assume an additive error model where the individual classifiers output, for each class w_k, a function of the features x such that $f_k(x) = P(w_k|x) + \varepsilon_k(x)$. The predictions are now made based on the output of the classifier instead of the actual posterior probabilities. Tumer and Ghosh consider the case in which the effect of using the predictions is a shift in the boundary by a certain amount b. This results in the probability of error being larger than the Bayes error by a certain amount they call the *added error*. An expression for the expected value of the added error is computed assuming that the ε_k are random variables with a certain mean and variance independent of the value of x, that they are uncorrelated across classes, that the b is small such that first-order approximations can be used and that the posterior probabilities are monotonic around the decision boundaries. Using the expression derived for the added error of a single classifier, they derive the corresponding expression for a classifier formed by the average estimates obtained from a set of N individual classifiers. Their final expression depends monotonically on the pairwise correlations between the error functions of the individual classifiers, supporting the intuition that good combination performance can be achieved if the classifiers being combined have complementary information such that when one classifier makes a mistake, some other classifier has the right answer. Fumera and Roli generalize Tumer and Ghosh's work by allowing the combination to be a weighted average instead of simple average and arrive at an optimization problem for the

combination weights. This problem is solved explicitly for the case of unbiased and uncorrelated estimation errors.

Two assumptions made by Tumer and Ghosh, as well as Fumera and Roli, are not well suited for the problem of interest in this paper but can be easily replaced for other assumptions without much consequence on the theory: the use of the classification error as measure of performance and the uncorrelatedness of the estimation errors across classes. Classification error is usually considered inappropriate for problems in which the prior probabilities of the classes are very different (as is the case in most speaker verification applications). For these cases, cost functions are usually defined where each type of error is assigned a different cost and the expected value of the cost function is used as performance measure. Derivations in Tumer and Ghosh (1996) can be easily modified to allow for cost functions. The Bayes classifier would now choose the class for which the loss is minimized instead of the posterior probability maximized. As for the estimation errors, the assumption of zero correlation can be easily relaxed when only two classes are considered. In fact, in our case we can assume that only one of the posteriors is estimated and the other one is calculated so that the sum is equal to one. In that case, the estimation errors would also sum to one. Hence, only one estimation error stays in the derivations and the zero correlation assumption is not needed.

Even though the results described above could be considered enough motivation for our development of the anticorrelation method in Section 4, in the next section we propose a different development that we believe presents an interesting alternative for the above framework. First, we do not use the additive error model with fixed bias and variance for the posterior probability estimates. In binary problems like speaker verification one can train classifiers that output a score instead of a probability. This score can be, for example, the output of a support vector machine or the logarithm of the ratio between the class likelihoods. These scores are assumed to be monotonically related to the posterior probability of one of the two classes and, hence, could be easily converted into this posterior. Nevertheless, in our speaker verification experiments we have found that combining the scores directly leads to better results than combining the estimated posteriors when using a linear combiner. In a combination experiment of all pairs of systems from a set of 13 systems, an average relative gain of 7.6% was found when combining scores versus posterior probabilities (obtained by learning a logit mapping of the scores) using linear logistic regression. The best performance across all combinations including any number of systems is obtained when combining the scores of six systems and this combination is 12% better than the best combination of posterior probabilities. Considering these results, we eliminate the assumption that the output of the classifiers is a posterior probability, allowing it to be a general continuous value (score). We then replace the additive error assumption with an assumption that the distribution of the scores for each class is Gaussian, which, as we will see, is a good approximation for most speaker verification scores.[2] Finally, we focus on a different measure of performance, the equal error rate, which is widely used on binary classification problems for its simplicity and for being independent of the class priors. This measure of performance depends on the overall shape of the posterior probabilities and not just on the difference between them, as is the case for the probability of error or the expected cost. Hence, no simple extension of the Tumer and Ghosh derivations could be made for this measure even if we

2. We believe this is the reason for the scores combining better with a linear combiner than the mapped posterior probabilities since when class distributions are Gaussian, the Bayes classifier is linear. Hence, the linear combiner is a good choice when combining speaker verification scores. The Gaussian approximation, on the other hand, is not good after the scores have been mapped to posterior probabilities making a linear combiner on the posterior probabilities suboptimal.

were willing to consider the system output to be a probability. Under these assumptions we arrive at an explicit expression for an upper bound on the optimal EER of the combination that is a function of the EER of the ensemble members and the pairwise class-conditional correlations.

2.2 Creating Diversity

The work described above, including the development we will show in Section 3, defines ways of quantifying the diversity in the ensemble and how this diversity affects the performance of the combined classifier under different sets of assumptions, but does not describe ways of achieving this diversity. Brown et al. (2005a) present a taxonomy of the methods for creating diversity found in the literature as of 2005. They divide the methods into implicit and explicit, where implicit methods are those that try to create diversity using randomization methods, while explicit methods do the same by directly taking into account some measure of the diversity being achieved. They further divide the methods into three groups based on how they affect the learners to create diversity: (1) by modifying the starting point in the hypothesis space, (2) by changing the set of accessible hypotheses or, (3) by defining how the hypothesis space is traversed. Modifying the starting point in the hypothesis space is applicable only for learners for whom the final hypothesis reached depends on some random initialization component, as is the case for neural networks. SVMs, on the other hand, do not fall into this category. Bagging (Breiman, 1996) and boosting (Freund and Schapire, 1997) are diversity creation methods of type 2, since each member of the ensemble is obtained by changing the training data, resulting in a different set of accessible hypotheses. In the case of bagging, the method is implicit since the training data for each ensemble is chosen randomly from the original set, while in the case of boosting, the training samples are weighted when training a new member of the ensemble in a way that ensures diversity, making it an explicit method of diversity creation. Allowing each member of the ensemble to use only a subset of features falls into the type 2 methods (Oza and Tumer, 2001). Finally, type 3 methods are those that directly aim at improving diversity by including a term measuring diversity in the objective function of the learner. The negative correlation (NC) learning algorithm is a notable example in this category (Liu, 1999; Rosen, 1996).

The goal of NC Learning is to minimize the squared error of an ensemble output computed as the average of the individual outputs in a regression context. This is done by adding a penalty term in the objective function of each individual neural network forming the ensemble. This penalty was shown (Brown et al., 2005b) to directly control the covariance term in the bias-variance-covariance trade-off. Zanda et al. (2007) extend the NC learning framework to classification problems by reinterpreting the Tumer and Ghosh model in a regression context. Our proposed anticorrelation method follows the spirit of the NC learning technique of explicitly creating diversity through the modification of the learner's objective function when the learners are SVMs instead of neural networks.

Much of the work on diversity creation has focused on the generation of large ensembles, where the number of classifiers is part of the design choices. The size of the ensemble is in fact another variable to be optimized. In this paper, we constrain the ensemble to contain a relatively small number of systems. These systems are different in nature, using different sets of features and different modeling methods and can be quite complex. Each of them might have been developed over several months or years of research. Hence, we do not take the size of the ensemble as a variable. The individual systems are fixed before hand, and all we are allowed to do is (perhaps only slightly) modify the training procedure of one or more of them in order to increase the diversity of the ensemble. The

speaker verification system described in this paper can, in fact, be considered as a cascade of two diversity creation methods. Given an enormous set of highly heterogeneous features coming from a few different sources in the speech signal (for example, prosodic or spectral information), we train separate classifiers for each type of feature, perhaps also varying the type of classifier used. This can be seen as a type 2 diversity creation method. The second stage of diversity creation focuses on making each new classifier added to the ensemble as new as possible. To achieve this we use a type 3 diversity creation method, where we add a penalty term in the SVM objective function, which explicitly aims at reducing the correlation between the new classifier and the preexisting ensemble. The proposed method is shown to be equivalent to defining a new kernel that we call the *anticorrelation kernel*.

Kocsor et al. (2004) introduced a method called *margin maximizing discriminant analysis* (MMDA) to obtain sucessive, mutually orthogonal, SVMs for a certain feature vector. In this method, presented as a nonparametric extension of linear discriminant analysis, the SVM optimization problem is modified by adding an orthogonality constraint with respect to the weight vectors from previous SVMs. The MMDA method can be seen as a particular case of the one proposed in this paper when the penalty coefficient is set to infinity. Furthermore, the constraints used here are more general, reducing to the ones used in MMDA when all systems in the ensemble are SVMs, use the same input features, and these features have identity within-class covariance matrices. Kernel-based methods for ensemble systems have also been used, for example, in Pavlidis et al. (2002) and Lanckriet et al. (2004). Pavlidis et al. compare three methods for combination of heterogeneous information for gene function detection: early, intermediate and late integration. Early integration (what we here call *feature-level combination*) consists of concatenating the features from the two different information sources into a single feature vector. Intermediate integration performs the combination at the kernel level, and late integration performs the combination at the last stage. This is what we are calling *score-level combination*. No explicit attempt at increasing diversity is made in the paper. Lanckriet et al. (2004) propose a method for combining kernel classifiers by learning a new kernel matrix that is a linear combination of the kernel matrices of the classifiers in the ensemble. This method requires prior knowledge of the test data, since instead of learning a function, they learn the labels on the set of unlabeled samples. This particular scenario is not applicable to speaker recognition where the speaker models are usually trained before any test sample is available.

2.3 System Combination

Once a diverse ensemble has been trained, the output of the individual classifiers must be combined into a single decision or score. Linear combiners are the most widely used methods for fusion of classifier outputs. In many cases the weights of the linear combination are determined based on the classification performance of the classifiers. For a survey of these methods, see Kuncheva (2004), Chapter 5. Obtaining the weights by training a "supra" classifier or combiner that takes the output of the individual classifiers as input is in many cases a better option. The main problem with this approach is that using the output of the individual classifiers on the data used for training them as training data for the combiner may result in suboptimal performance. The stacked generalization method (Kuncheva, 2004, Chapter 3) can be used in these cases to generate the training data for the combiner.

The focus in this paper is on diversity creation rather than on the methods used for combining the ensemble. Hence, we choose a simple but effective combination method: a linear supra-classifier

trained with data generated by the stacked generalization method using logistic regression. Linear logistic regression was shown in our previous work (Ferrer et al., 2008a) to perform comparably to or better than other linear and nonlinear classifiers for the combination of speaker verification scores, and it is one of the most commonly used methods for combining speaker verification systems (Brummer et al., 2007). Other common combination procedures used in speaker verification include neural networks (Reynolds et al., 2003; Ferrer et al., 2006), support vector machines (SVM) (Ferrer et al., 2006), and weighted summation using empirically determined weights (Dehak et al., 2007).

3. A Simple Model for System Combination

Consider a binary classification task with classes $y \in \{a,b\}$ for which N separate classifiers are available. In speaker verification, class a corresponds to *impostor* and b to *true-speaker*. Classifier i produces a score, f_i, which can be thresholded to obtain the final decision. That is, the estimated class $\hat{y}$ is given by

$$\hat{y} = \begin{cases} a & \text{if } f_i < t_i \\ b & \text{if } f_i \geq t_i, \end{cases} \tag{1}$$

where t_i is a tunable threshold.

Here, we consider a setup where the scores from the individual classifiers are combined into a single score $f_c = f_c(f_1, \ldots, f_N)$. This final score is the one that is later thresholded to obtain the estimated class for the sample. The goal is then to optimize the performance of the final combination, not that of the individual systems.

In this section we develop a simple model for this problem, which will lead us to an intuitive conclusion about what could be done to improve the final performance when training a new system for combination with others.

3.1 Mahalanobis Distance as a Surrogate for EER

Consider for now a single score f, corresponding to a random variable, F.[3] A usual way of measuring performance of a score when Equation (1) is used to estimate the class of the samples is equal error rate (EER), the false acceptance rate when the false rejection and false acceptance rates are equal. In speaker verification, a false acceptance (which we will call $e_{b|a}$) is an impostor trial accepted by the system as the target speaker, and a false rejection ($e_{a|b}$) is a true-speaker trial considered an impostor trial by the system.

If we make the assumption that the conditional distribution of the scores for each class is Gaussian, we can obtain an explicit bound for the EER as a function of the Mahalanobis distance between the class-conditional means. Figure 1 shows how to calculate the EER for this case. We assume that the class-conditional distribution of the scores is given by

$$F \mid Y = y \sim \mathcal{N}(\mu_y, \sigma_y^2), \text{ for } y = a,b,$$

where Y is the random variable corresponding to the trial's class. We further assume, without loss of generality, that $\mu_b \geq \mu_a$, so that (1) is the best way of assigning the labels for a given threshold.

3. Throughout the paper we will adopt the notation of using lower case letters for the samples of the random variable noted by the corresponding capital letter.

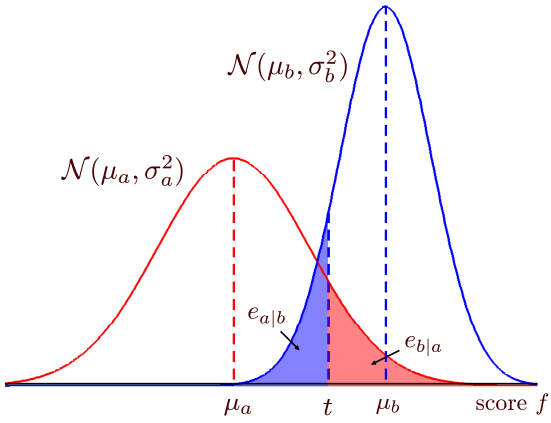


Figure 1: EER calculation when the class-conditional distributions are Gaussian. The EER is equal to $e_{a|b}$ when t is chosen such that $e_{a|b} = e_{b|a}$.

With these assumptions we can compute $e_{a|b}$ and $e_{b|a}$ for a certain value of the threshold t as

$$e_{a|b}(t) = \phi\left(\frac{t-\mu_b}{\sigma_b}\right), \tag{2}$$

$$e_{b|a}(t) = 1-\phi\left(\frac{t-\mu_a}{\sigma_a}\right), \tag{3}$$

where ϕ is the cumulative distribution function of the standard normal distribution $\mathcal{N}(0,1)$.

The EER is given by $e_{b|a}(t^*)$ when t^* is chosen such that $e_{b|a}(t^*) = e_{a|b}(t^*)$. In the appendix we prove the following upper bound on the EER:

$$\text{EER} = e_{b|a}(t^*) = e_{a|b}(t^*) \leq \phi(-\frac{1}{2}\frac{\delta\mu}{\sigma}), \tag{4}$$

where $\delta\mu = \mu_b - \mu_a$, and σ is any value that satisfies $\sigma \geq (\sigma_a + \sigma_b)/2$. Equality is achieved for $\sigma = (\sigma_a + \sigma_b)/2$, but this value of σ does not result in a nice expression later on, when we want to use it to optimize the combination performance. On the other hand,

$$\sigma = \sqrt{(\sigma_a^2 + \sigma_b^2)/2}$$

also satisfies the inequality and does result in a nice expression that we can later use. In the rest of the paper we will use

$$\mathcal{M}^2 = \frac{2\delta\mu^2}{\sigma_a^2 + \sigma_b^2} \tag{5}$$

as a surrogate for the EER of the system. $\mathcal{M}$ is the Mahalanobis distance between two Gaussian distributions with distance between the means $\delta\mu$ and variance $(\sigma_a^2 + \sigma_b^2)/2$. Using (4), we get that

$$\text{EER} \leq \phi(-\frac{\mathcal{M}}{2}).$$

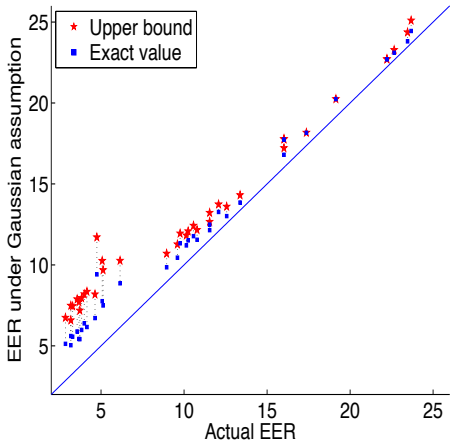


Figure 2: Actual EER for a set of 37 systems versus the upper bound and the exact value under Gaussian assumptions.

Our strategy in the remainder of the paper will be to reduce the value of the upper bound, with the hope that this will result in a reduction of the EER. Since ϕ is monotonically increasing in its argument, decreasing the value of $-\mathcal{M}$ (or increasing the value of $\mathcal{M}$) will decrease the value of the upper bound.

Figure 2 shows a scatter plot of the actual EER for a set of 37 different individual systems using different features or different modeling techniques (some of these systems are described by Ferrer et al. (2006)) versus the upper bound on the EER under Gaussian assumption given by $\phi(-\delta\mu/\sqrt{2(\sigma_a^2+\sigma_b^2)})$ and the exact value of the EER, also under Gaussian assumption, given by $\phi(-\delta\mu/(\sigma_a+\sigma_b))$. We can see that the upper bound is tight for high EER values. When the difference between the class-conditional covariance for both classes is large (which, in our experiments, is the case for the better systems), the upper bound becomes looser. We see that the exact estimation under the Gaussian assumption performs significantly better in this case. The remainder of the error is due to the inaccuracy of the Gaussian assumption. Figure 3 shows the actual distribution of two of the systems used in this paper compared to their Gaussian approximation. Figure 3.a corresponds to one of the low EER systems that drifts away from the diagonal in Figure 2, while 3.b corresponds to a system with EER around 12%. We can clearly see that the Gaussian approximation in this case is inaccurate, which in turn explains the inaccuracy of the upper bound and the exact formula developed in this section. Interestingly, even when the Gaussian approximation is inaccurate, (4) seems to still hold. Our procedure of trying to minimize this upper bound in the hope that the actual EER will be pushed down would then still be valid.

3.2 Maximum Mahalanobis Distance Combination

Returning to the problem of combining the output of several subsystems into a single score, we use the results from the previous section and maximize (5) to find the optimal parameters of the combiner. This will result in a simple formula that can predict the performance of the combination of

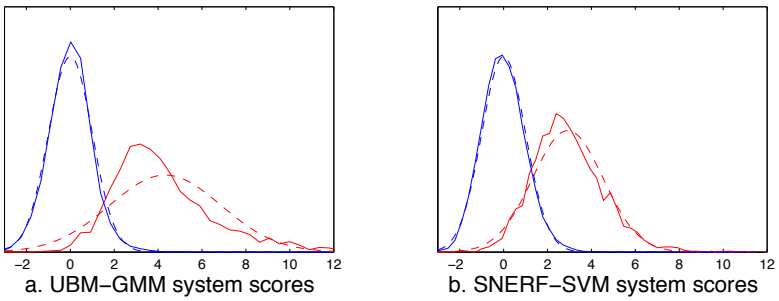


Figure 3: Distribution of scores for two individual systems compared to their Gaussian approximation.

several subsystems given their individual EERs (or $\mathcal{M}$s) and the average class-conditional covariance matrix for the vector of scores. To do this we further assume that the combination is performed with a linear function of the individual scores. This is not a very restrictive assumption for speaker verification since for this task we have repeatedly found that linear combination procedures perform as well (or better) than nonlinear ones (Ferrer et al., 2008a). The form of the combined scores as a function of N individual scores is then

$$f_c = \sum_{i=1}^{N} \alpha_i f_i = \alpha^t f,$$

where $\alpha = (\alpha_1 \dots \alpha_N)^t$ is the vector of weights and $f = (f_1 \dots f_N)^t$ the vector of individual scores. We assume that the class-conditional distribution of the f_i's is jointly Gaussian. Hence, each of the individual scores and the combined score satisfy the assumptions made in the previous section. The class-conditional distributions of F_c, the random variable corresponding to the combined score, are given by

$$F_c \mid Y = y \sim \mathcal{N}(\alpha^t \mu_y, \alpha^t \Sigma_y \alpha), \text{ for } y = a, b,$$

where $\mu_y = (\mu_{y1} \dots \mu_{yN})$ is the vector of means and $\Sigma_y = E[(F - \mu_y)(F - \mu_y)^t | Y = y]$ the covariance matrix for class y for random variable F corresponding to vector f. We can now compute $\mathcal{M}^2$ (Equation 5) for the combined score as a function of the parameters α:

$$\mathcal{M}^2 = \frac{2(\alpha^t \mu_b - \alpha^t \mu_a)^2}{\alpha^t \Sigma_a \alpha + \alpha^t \Sigma_b \alpha} = \frac{\alpha^t \Delta \Delta^t \alpha}{\alpha^t \Sigma \alpha}, \tag{6}$$

where $\Delta = \mu_b - \mu_a$ and

$$\Sigma = 1/2(\Sigma_a + \Sigma_b). \tag{7}$$

We wish to find the α that maximizes this expression. Define $\beta = \Sigma^{1/2}\alpha$, where $\Sigma^{1/2}$ is a symmetric matrix square root of Σ (which can be computed from the eigendecomposition of Σ, which exists and can be chosen to be symmetric since Σ is symmetric). Replacing this in (6), and using the Cauchy-Schwarz inequality,

$$\begin{aligned} \mathcal{M}^2 &= \frac{\beta^t \Sigma^{-1/2} \Delta \Delta^t \Sigma^{-1/2} \beta}{\beta^t \beta} = \frac{\|\Delta^t \Sigma^{-1/2} \beta\|^2}{\|\beta\|^2} \\ &\leq \|\Delta^t \Sigma^{-1/2}\|^2 = \Delta^t \Sigma^{-1} \Delta, \end{aligned}$$

with equality when $\beta \propto \Sigma^{-1/2}\Delta$. Hence, the optimal α is a vector in the direction of $\Sigma^{-1}\Delta$.[4]

Note that we have arrived at the definition of the Mahalanobis distance in multiple dimensions ($\Delta^t\Sigma^{-1}\Delta$), which is an intuitive result. We now know that the EER of the combination of the individual systems will be at most $\phi(-\sqrt{\Delta^t\Sigma^{-1}\Delta}/2)$. If we can devise a way of increasing $\Delta^t\Sigma^{-1}\Delta$, this upper bound will decrease.

Anderson and Bahadur (1962) consider the problem of finding all the admissible linear procedures for classifying into two multivariate normal distributions. They show that $\alpha = (t_a\Sigma_a + t_b\Sigma_b)^{-1}\Delta$ corresponds to an admissible procedure for any t_a and t_b (that is, no other linear procedure will have, simultaneously, strictly better $e_{b|a}$ and $e_{a|b}$) as long as $t_a\Sigma_a + t_b\Sigma_b$ is positive definite. Unfortunately, the values for t_a and t_b that correspond to the EER have to be numerically obtained. No explicit expression is available for the general case. Since our purpose here is to obtain a simple explicit expression for the EER (as a function of the EERs of the individual systems and some other parameters) that we can use to analyze the problem, we have settled for a bound on the EER instead of using the implicit expression developed by Anderson and Bahadur.

3.3 Analysis for Combination of Two Systems

We now focus on the case $N = 2$. In this case, we have that the optimal $\mathcal{M}^2$ ($\Delta^t\Sigma^{-1}\Delta$) is given by

$$\begin{aligned}\mathcal{M}^2 &= \frac{\delta_1^2\sigma_{22} + \delta_2^2\sigma_{11} - 2\delta_1\delta_2\sigma_{12}}{\sigma_{11}\sigma_{22} - \sigma_{12}^2} \\ &= \frac{\mathcal{M}_1^2 + \mathcal{M}_2^2 - 2\rho\mathcal{M}_1\mathcal{M}_2}{1-\rho^2},\end{aligned} \tag{8}$$

where δ_i is component i of vector Δ, σ_{ij} is component ij of matrix Σ, $\mathcal{M}_1 = \delta_1/\sqrt{\sigma_{11}}$ and $\mathcal{M}_2 = \delta_2/\sqrt{\sigma_{22}}$ are the Mahalanobis distances for the individual systems, and

$$\rho = \sigma_{12}/\sqrt{\sigma_{11}\sigma_{22}}. \tag{9}$$

Note that ρ is *not* the correlation between the two systems in the usual sense, since Σ is not the covariance matrix of F, but the average class-conditional covariance.

Let us call the upper bound on the EER, $\hat{e}$, that is, $\hat{e} = \phi(-\mathcal{M}/2)$. Figure 4 shows some curves of $\hat{e}_c$, the upper bound on the performance of the combination for two systems, as a function of ρ. To create these plots we take two actual systems (the MLLR-SVM and the SNERF-SVM, as described in Section 6.2) and compute the upper bound on the EERs based on their Gaussian approximation. We also compute matrix Σ and from there, ρ. This gives a single point in this graph (marked with a star). We can now vary ρ, keeping $\hat{e}_1$ and $\hat{e}_2$ (the upper bound on the performance of the individual systems) fixed, to obtain a curve. We can also obtain curves for different values of $\hat{e}_2$. These curves show us the relation between the performance of system 2, the value of ρ between the two systems, and the performance of their combination. We can see that for this particular set of systems, we could degrade the second system 100% (that is, from 15.6% to 31.3%) and still get a gain in the

4. Note that this is the same direction one would obtain with linear discriminant analysis (LDA). As mentioned in the introduction, for the experiments in this paper we will use linear logistic regression (LLR) to train the combination weights. The reason for using LLR instead of LDA, despite the result obtained in this section, is that the results in this section are obtained under the assumption of class-conditional Gaussianity. When Gaussianity is not closely satisfied, LLR is believed to be a safer choice being more robust to outliers (Hastie et al., 2001, Section 4.4). In fact, a set of initial experiments showed that LLR was significantly better than LDA on our speaker verification data.

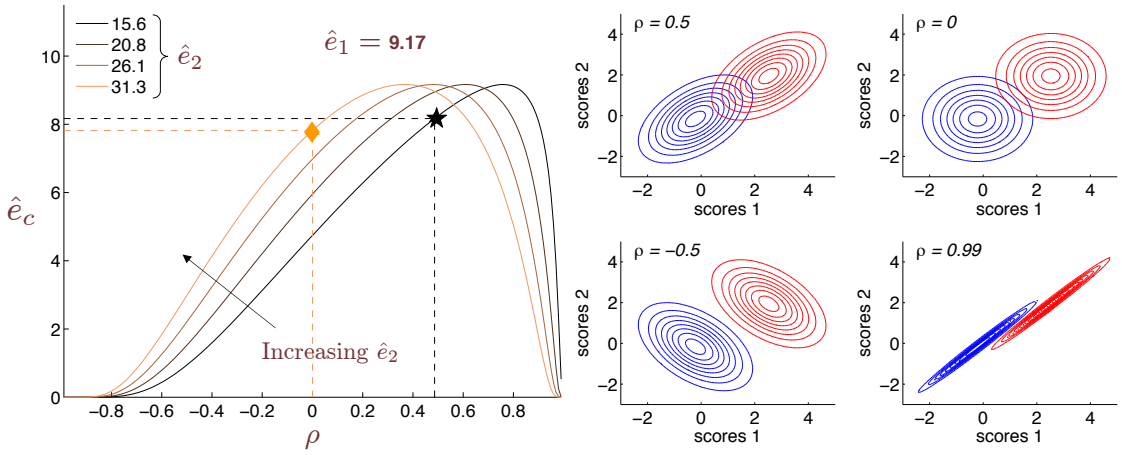


Figure 4: Left: Curves of $\hat{e}_c$ (the upper bound on the EER of the combination) for two systems as a function of ρ, for $\hat{e}_1$ fixed and various values of $\hat{e}_2$, where $\hat{e}_i$ denotes the upper bound on the EER of system i. Right: Simulated contour plots for the scores of two systems assuming class-conditional distributions are Gaussian with equal covariance matrix. Marginal distributions are kept fixed for all four figures, only ρ is changed. These plots correspond to four different points in the darker curve from the left plot.

performance of the combination (or at least in the upper bound) over what we get with the original pair of systems if, at the same time, we were able to decrease the ρ from 0.48 to 0 (this point is denoted by a diamond in the graph).

At first sight, these curves may seem to go against intuition, since when ρ increases, after reaching a peak (occurring at the minimum of $\mathcal{M}_1/\mathcal{M}_2$ and $\mathcal{M}_2/\mathcal{M}_1$), they go down again. That is, very high values of ρ result in extremely good combination performance. Similarly, when ρ turns negative, the combination performance improves, reaching zero EER for $\rho = -1$. All these cases can be easily understood using contour plots of the scores from two systems for varying values of ρ. The right plot of Figure 4 shows four different cases. Here we keep the marginal distributions of the systems fixed and vary only ρ, which implies that the performance of the two individual systems stays fixed for the four different plots. That is, the four plots correspond to four different points in a single curve like the ones in the left plot. Furthermore, we set the covariance matrix between the two systems to be the same for both classes. In this way, the upper bound (4) is exact and ρ is the within-class correlation (assuming both classes have the same prior). We can see that the separation between the two classes is highly dependent on the value of ρ. The first plot shows a typical case, where $\rho = 0.5$. The second plot shows the case of $\rho = 0$. We can already see that the overlap between the two classes has been significantly reduced, even though the marginal distributions have not changed. The third plot shows the case of negative ρ. This implies that both systems produce errors in a negatively correlated way, which makes the combination of those two systems extremely effective at reducing the error rate. The fourth plot illustrates the case in which ρ is larger than the value for which the $\hat{e}_c$ curve peaks, showing good separation between the two classes. Perfect classification with $\rho = 1$ is possible only when both classes have exactly the same

covariance matrix except possibly for a scalar factor (so that the contour plots become parallel lines) and the mean vectors for both systems are different. This case corresponds to a zero value in the denominator of (8) and a nonzero at the numerator.

In practice, if we take pairs of all 37 systems plotted in Figure 2 and compute the actual ρ and the ρ at the peak (that is, $\rho_{peak} = \min(\mathcal{M}_1/\mathcal{M}_2, \mathcal{M}_2/\mathcal{M}_1)$) we find that, on average, the actual ρ is 0.26 to the left of ρ_{peak}. Considering this empirical fact it seems unreasonable to try to increase ρ in order to improve the combination performance, since it would require a large increase in order to go past the peak into an area where the combination performance is better than the original. Furthermore, this would work only if the class-conditional covariance matrices were equal, which is usually not the case. Hence, in this paper, our strategy will be to try to decrease the value of ρ. Of course, we also require that the performance of each individual system stays reasonably close to its original performance, which we hope will result in a new point (in a plot like the one in the left of Figure 4) located toward the left and lower than the original point. In the next section we introduce our method for achieving this goal.

4. Anticorrelation Kernel

Suppose that two separate classifiers S and B are available, where S is required to be an SVM, but B can be any classifier that produces a score for each sample. We will consider B to be a black box from which we have only the scores that it produces. As we have been assuming, the final classification decision will be made based on a combination of the outputs generated by both classifiers. Our strategy will be to train system S using information about system B in order to improve the combination performance over the one obtained when system S is designed with no knowledge of system B.

4.1 Support Vector Machines

Consider a labeled training set with m samples, $T = \{(x_j, y_j) \in \mathcal{R}^d \times \{-1,+1\}; j = 1,...,m\}$, where x_j is the feature and y_j the class corresponding to sample j. The goal is to find a function $f(x) = w^t x + c$, such that $\text{sign}(f(x))$ is the predicted class for feature vector x. The standard (primal) SVM formulation for classification is given by Vapnik (1999):

$$\begin{array}{lll} \text{minimize} & J(w,\varepsilon) = \dfrac{1}{2} w^t w + C \displaystyle\sum_{j=1}^{m} \varepsilon_j & \\ \text{subject to} & y_j(w^t x_j + c) \geq 1 - \varepsilon_j & j = 1,...,m \\ & \varepsilon_j \geq 0 & j = 1,...,m. \end{array} \tag{10}$$

Minimizing the norm of the weight vector is equivalent to maximizing the margin between the samples and the hyperplane. The *slack* variables ε_j allow for some samples to be at a distance smaller than the margin to the separating hyperplane or even on the wrong side of the hyperplane. The parameter C controls the trade-off between the size of the margin and the total amount of error. By deriving the dual form of the optimization problem above we find that input vectors appear only as inner products with each other. Hence, if we wish to transform the input features with a certain function $\phi(x)$ we only need to be able to compute the inner products between the transforms for any pair of samples, that is, we only need to know the function $K(x_k, x_l) = \phi(x_k)^t \phi(x_l)$. This fact is what allows for complex transforms of the input features to be used, as long as the kernel

function $K(x_k, x_l)$ can be easily computed. In the next section we will develop the proposed method considering an inner product kernel. The general case will be treated in Section 4.5.

The above setup corresponds to a classification problem. The regression problem can also be posed as a convex optimization problem by choosing an appropriate distance measure (Vapnik, 1999; Smola and Schölkopf, 1998) with the objective function given by the sum of the square norm of the weight vector and an error term, as in the classification case. The dual of this problem again takes a form in which features appear only in inner products with other features, which again allows for the kernel trick to be used. Hence, even though the derivations in this paper will be done considering a classification problem for simplicity, the method described and the interpretations given can be equally applied to SVM regression problems.

4.2 Modified Support Vector Machines

As we saw in Section 3.3, reducing the ρ value between system S and system B could lead to an improvement in combination performance as long as the performance of the individual systems does not degrade too greatly. We propose to add a term in the SVM objective function ($J(w, \varepsilon)$ in Equation (10) that introduces a cost for a model that results in high value of ρ. Ideally, we would like this term to be given by $\lambda\rho$ so that low values of ρ are encouraged, including negative ones. Unfortunately, this term would make the new objective function nonconvex, making the optimization problem much more complex. To see this, let us derive an expression for ρ as a function of the SVM weights. Let $S = w^t X + c$, where w and c are the SVM parameters. We can compute $\sigma_{12} = \sigma_{SB}$ (component $1,2$ of Equation 7) as[5]

$$\sigma_{SB} = \frac{1}{2} \sum_{y=\{a,b\}} E[(B-\mu_{b,y})(S-\mu_{s,y})|Y=y] = w^t K,$$

where we are using the notation $\mu_{v,y} = E[V|Y=y]$ for any random variable V (scalar or vectorial), and where

$$K = \frac{1}{2} \sum_{y=\{a,b\}} E[(B-\mu_{b,y})(X-\mu_{x,y})|Y=y]. \tag{11}$$

K is simply the vector of average class-conditional covariances between each input feature and the scores from system B. The value of vector K can be estimated from the training set T as

$$\tilde{K} = \frac{1}{2} \sum_{y=\{a,b\}} \frac{1}{m_y} \sum_{j|y_j=y} (b_j - \tilde{\mu}_{b,y})(x_j - \tilde{\mu}_{x,y}), \tag{12}$$

where b_j is the score generated by system B for sample j, m_y is the number of samples in T from class y, $\tilde{\mu}_{b,y} = \frac{1}{m_y} \sum_{j|y_j=y} b_j$, and $\tilde{\mu}_{x,y} = \frac{1}{m_y} \sum_{j|y_j=y} x_j$.

Similarly, we can compute $\sigma_{11} = \sigma_{SS}$ (component $1,1$ of Equation 7) as $w^t M w$, where M is the average class-conditional covariance matrix for the feature vector X; and $\sigma_{22} = \sigma_{BB}$ (component $2,2$ of Equation 7) as the average class-conditional variance for the B scores that we will call v. We can now write ρ^2 as

$$\rho^2 = \frac{\sigma_{SB}^2}{\sigma_{SS}\sigma_{BB}} = \frac{w^t K K^t w}{v\, w^t M w}.$$

5. We use B and S to refer to the systems and the random variables corresponding to the scores produced by these systems. The actual meaning should be clear from the context.

This expression and its square root are nonconvex functions of w. Hence, adding the term $\lambda\rho$ to the objective function of the SVM problem would make the optimization problem nonconvex. On the other hand, the numerator of ρ^2, σ_{SB}^2 is a convex function of w and, using it as a regularization term results in a new problem that is equivalent to a standard SVM problem with a new kernel function. To see this, write

$$\begin{aligned} J_\sigma(w,\varepsilon) &= \frac{1}{2}w^t w + \frac{\lambda}{2}w^t KK^t w + C\sum_i \varepsilon_i \\ &= \frac{1}{2}w^t Aw + C\sum_i \varepsilon_i, \end{aligned}$$

where $A = I + \lambda KK^t$ is a symmetric positive semidefinite matrix. We can now change variables $\tilde{w} = Bw$, with B symmetric and $B^tB = A$ (i.e., B is a matrix square root of A and, since A is a positive definite symmetric matrix, it always exists and can be chosen to be real and symmetric) and write it as

$$\begin{aligned} &\text{minimize} && J_\sigma(\tilde{w},\varepsilon) = \tfrac{1}{2}\tilde{w}^t\tilde{w} + C\textstyle\sum_j \varepsilon_j \\ &\text{subject to} && y_j(\tilde{w}^t z_j + c) \geq 1 - \varepsilon_j && j = 1,\ldots,m \\ &&& \varepsilon_j \geq 0 && j = 1,\ldots,m, \end{aligned}$$

where

$$z_j = B^{-1}x_j. \tag{13}$$

We see that the appropriate choice of regularization term led us to a very simple new optimization problem. The disadvantage of this choice is that it does not *directly* achieve our goal of minimizing ρ. For example, negative values of ρ corresponding to negative values of σ_{SB} will generally not be encouraged by the new objective function because the maximum margin (that is, the minimum value of w^tw) corresponds, in most practical cases, to positive values of σ_{SB}. Hence, in practice, for each negative value of σ_{SB} the corresponding positive one will result in a smaller value of the objective function and will be preferred to the negative one. Furthermore, minimizing σ_{SB} does not imply minimization of ρ, even for positive values, since the denominator is not being taken into account. Nevertheless, we empirically find that the new optimization problem achieves its goal of reducing ρ. In particular, when λ is large, σ_{SB} is pushed toward zero, forcing ρ to become zero.

Directly finding the matrix B^{-1} in (13) is computationally expensive since in general the dimension d of the feature vectors x_i can be very large and the matrix B is a full matrix of size $d \times d$. Nevertheless, since matrix A has a very particular structure, we can find an expression for its inverse using the matrix inversion lemma, by which $A^{-1} = I - \frac{\lambda}{1+\lambda K^tK}KK^t$. Hence, one way of implementing the proposed method is to define a kernel $K(x_k,x_l) = x_k^tB^{-t}B^{-1}x_l = x_k^tA^{-1}x_l$ to be used by the SVM. This kernel satisfies the mercer conditions (i.e., it is a valid kernel) since A is a positive semidefinite matrix. Using the expression for A^{-1} above we get

$$K(x_k,x_l) = x_k^tx_l - \frac{\lambda}{1+\lambda K^tK}x_k^tKx_l^tK. \tag{14}$$

We call this kernel the *anticorrelation kernel*. The computation of $K(x_k,x_l)$ in Equation (14) requires only the calculation of three inner products, which makes the method computationally feasible. Another approach is to transform directly the features using (13). This is also computationally feasible because the inverse of the matrix B has a simple expression. To find this expression we first

note that the matrix B^{-1} is a matrix square root of A^{-1}. Hence, we need to find a matrix that when multiplied by its transpose results in $A^{-1} = I - \frac{\lambda}{1+\lambda K^t K} K K^t$. It is easy to show that

$$B^{-1} = I - \frac{\alpha}{K^t K} K K^t$$

satisfies this condition when $\alpha = 1 \pm \frac{1}{\sqrt{1+\lambda K^t K}}$. Hence, given a certain value for λ we can find the corresponding α and transform each feature vector using (13). This means that we can implement the proposed method by transforming the input features using the following expression:

$$z_i = x_i - \alpha \frac{K^t x_i}{K^t K} K. \tag{15}$$

In the case of speaker verification, a separate K vector is computed for each target model being trained. Hence, doing the transformation in the feature domain is inefficient, since there is not a single transformation for each feature vector x_i, but one for each target model. The kernel implementation might be preferable in this case. Furthermore, as we will see in Section 4.5, when the original SVM problem uses a kernel other than the inner-product one, implementing the anticorrelation method as a kernel may be the only feasible option.

4.3 Interpretation of the Modified Problem

To give an interpretation of the new SVM problem, we first need to understand the meaning of the direction given by the vector K. The average class-conditional covariance between the scores from system B and the scores from system S is given by $w^t K$. For a fixed value of $\|w\| = c$, the w that maximizes the absolute value of $w^t K$ is given by $w = cK/\|K\|$. Hence, K gives the direction for the vector w of SVM weights for which the average class-conditional covariance between the two systems is maximum. A w orthogonal to K would result in zero average class-conditional covariance between the two systems. The term $\|w^t K\|^2$ that we have added to the objective function of the SVM problem has the effect of penalizing any w vector with a large component in the direction of K. Our goal is to find a w as orthogonal to K as possible without degrading the performance of the system so much that the overall combination starts to degrade. This balance can be achieved by tuning the parameter λ.

We can interpret the kernel given by (14) in a similar way. When λ is small this kernel is close to the linear kernel. When λ grows to infinity the kernel subtracts the product of the projections of the points x_k and x_l into the vector K from the linear kernel. The resulting value of the kernel will be small if x_k and x_l are both aligned with K. Since the SVM will make an effort to separate only points from different classes that give a high kernel value (that is, that are more "similar"), this means that we consider vectors whose directions are close to that of K to be unimportant and, consequently, we emphasize the importance of the vectors whose directions are orthogonal from that of K. This results in a more effective usage of the features, ignoring those directions that would lead to high average class-conditional covariance between the systems and taking advantage of the rest.

Finally, if we choose to implement the method as a feature transform instead of a kernel function, the resulting features have a very simple interpretation. When $\lambda = \infty$, Equation (15) becomes $z_j = x_j - \frac{K^t x_j}{K^t K} K$, which is the expression for subtracting from x_j its component on direction K. If λ is not ∞ then only a part of the component is subtracted.

4.4 Extension for Multiple Preexisting Scores

An extension to the presented method can be considered where N previous systems are available, $B_1,\ldots,B_N$, and we wish to train S to combine well with them. A generalization of the formulas above can be derived for this setup. We rewrite the objective function as

$$\begin{aligned} J_\sigma(w,\varepsilon) &= \frac{1}{2}w^t w + \sum_{k=1}^{N}\frac{\lambda_k}{2}w^t K_k K_k^t w + C\sum_i \varepsilon_i \\ &= \frac{1}{2}w^t A w + C\sum_i \varepsilon_i, \end{aligned}$$

where now A is given by $I+\sum_{k=1}^{N}\lambda_k K_k K_k^t$ and it is still positive definite and symmetric. The same approach used above can be used here to simplify the problem to a standard SVM problem. We can still use the inversion lemma by considering matrices

$$\begin{aligned} K &= [K_1 \ldots K_N], \\ \Lambda &= \mathrm{diag}(\lambda_1 \ldots \lambda_N), \end{aligned}$$

so that $I+\sum_{k=1}^{N}\lambda_k K_k K_k^t = I + K\Lambda K^t$. We can then use the lemma to get $A^{-1} = I - K(\Lambda^{-1} + K^tK)^{-1}K^t$. When $\lambda_k = \infty$ for all k, $A^{-1} = I - K(K^tK)^{-1}K^t$. This matrix is idempotent (and symmetric), hence $B^{-1} = A^{-1}$. The transformed features z_i for this case are then given by $z_i = x_i - K(K^tK)^{-1}K^t x_i$, which is the projection of x_i on the complementary space to that spanned by vectors K_1 through K_N.

4.5 Extension for General Kernels

The development on Section 4.2 was done using inner-product kernel SVMs as the starting point. In this section we show that the method can be implemented for any kernel function.

Consider a problem for which $K_0(x,y) = \phi_0(x)^t\phi_0(y)$ has been found to perform better than the inner-product kernel. One way of implementing the anticorrelation method in this case is to simply transform the features using $\phi_0(x)$ and then treat the transformed features as the feature vectors x in Section 4.2. This is conceptually simple, but could be extremely costly computationally if the dimension of $\phi_0(x)$ is large compared to the dimension of x, or impossible if the transformation ϕ_0 is infinite dimensional (as in the case of the Gaussian kernel). Luckily, there is a way of implementing the anticorrelation method without ever computing the transform but only the kernel function between pairs of features.

In Section 4.2 we found that one way of implementing the proposed method is by the use of the *anticorrelation* kernel, defined by Equation (14). In practice, vector K in that equation is computed from data. We call this empirical value for K, $\tilde{K}$ (Equation 12). We can write $\tilde{K}$ as a linear function of the features x_j used to compute it. That is, $\tilde{K} = \sum_j c_j x_j$, where the c_j depend on the m_y's and all the b_i's. If we now replace every x in Equation (14) by $\phi_0(x)$ and K by $\sum_j c_j\phi_0(x_j)$, we get

$$\begin{aligned} K(x_k,x_l) &= \phi_0(x_k)^t\phi_0(x_l) - \frac{\lambda\sum_j c_j\phi_0(x_j)^t\phi_0(x_k)\sum_j c_j\phi_0(x_j)^t\phi_0(x_l)}{1+\lambda\sum_j\sum_i c_j c_j\phi_0(x_i)^t\phi_0(x_j)} \\ &= K_0(x_k,x_l) - \frac{\lambda\sum_j c_j K_0(x_j,x_k)\sum_j c_j K_0(x_j,x_l)}{1+\lambda\sum_j\sum_i c_j c_j K_0(x_i,x_j)}. \end{aligned}$$

The anticorrelation kernel can then be computed exclusively as a function of the original kernel K_0. The processing time is now significantly increased, though, since two sums over the samples used to obtain K are needed every time the kernel is computed (the denominator in the second term can be precomputed and reused, since it does not involve x_k or x_l). The extension for multiple preexisting scores follows the same steps as above. In this paper, the inner-product kernel is used for all experiments.

4.6 Other Approaches

Our goal is to obtain the best possible combination performance given the available systems. The approach presented above is one path toward this goal. Two other ways of approaching this problem are considered here.

4.6.1 FEATURE-LEVEL COMBINATION

When system B is also an SVM system and the features corresponding to the samples used for training system S are also available for system B, an SVM using the features from both systems concatenated into a single vector can be trained. The resulting SVM is in itself a combination procedure, which, ideally, should make optimal use of the features from both systems. This may not be true in practice, though, since a larger feature vector increases the complexity of the system, making it more prone to overfitting the training data. A further refinement of this approach consists of weighting the vector components, assigning weight α to the features from one of the original systems and weight $1-\alpha$ to the other features. This is done by multiplying the components of the square-norm of w in the SVM objective function by the inverse of the corresponding feature weight. That is, we replace $\|w\|^2 = \sum_i w_i^2$ with $\sum_i w_i^2/\beta_i$, where $\beta_i = \alpha$ for the features from one set and $1-\alpha$ for the features from the other set. This allows us to compensate for different lengths in the original vectors or to bias the training procedure to make more use of the features from the better-performing system. Feature-level combination is usually costly and sometimes even infeasible, given the large size of the original feature vectors, and can be considered only if both systems being combined are SVM systems.

4.6.2 FEATURE+SCORE COMBINATION

Another method can be considered in which we present the scores generated by system B as input features to the SVM, along with all the features from system S. Again, a larger weight can be given to the component corresponding to the score from system B than to the features from S.

5. Experiments on Artificial Data

To test the proposed kernel on a simple task, we generated data for two classes with model

$$Z = C\tilde{Z} + m_Y,$$

where $\tilde{Z}$ is a vector of size d where the components are generated independently with normal distribution with zero mean and unit variance. C is a square random matrix intended to create correlation between the features. Its components are drawn from a uniform distribution, and a scaling factor is

applied to force the maximum variance of Z to be 1. The class-dependent mean vector is given by

$$m_Y = \begin{cases} (0\ldots0)^t & \text{if } Y = a \\ (m\ldots m)^t & \text{if } Y = b. \end{cases}$$

We take half of the features and train a linear SVM (inner-product kernel), which serves as system B. The remaining features are used to train system S starting with an inner-product kernel. The anticorrelation kernel is implemented for varying values of λ. We create two separate sets, one for training, with 900 examples of class a and N examples of class b, and one for testing, with 10 times more data than in the training set.

The combination is performed using a linear logistic regression model, trained on the training set with the scores from the two SVM systems, B and S, for each value of λ. Since the scores obtained on the training set are overly optimistic, we use 10-fold cross-validation on the training set to create the B and S scores used to train the combiner (Kuncheva, 2004, Section 3.2.2). The scores from cross-validation for system B are also used to estimate the vector $\tilde{K}$ as in (12). When only a few samples from one of the classes are available, the estimation of $\tilde{K}$ can be noisy. In our simulation we vary the number of samples available for class b, keeping the number for a fixed; hence, in order to keep the variance of the estimator stable across experiments, we use only samples from class a to estimate $\tilde{K}$.

Figure 5 shows the error rates for the test data for system S, system B, and the score-level combination as a function of the value of λ, for $m = 3.0, N = 900$ and $d = 250$. The figure also shows results for the feature-level and the feature+score combination procedures, explained in Section 4.6. The weight α for these two systems was tuned using 10-fold cross-validation on the training set. For these two cases and for system B, the error does not depend on λ. On the other hand, the value of ρ between S and B decreases with λ (reaching a value close to zero). We see that, in practice, ρ is effectively reduced as λ increases, even though we use $\lambda\sigma^2_{SB}$ as regularization term instead of $\lambda\rho^2$. The error for system S also varies with λ. The degradation in performance is expected since we are trading off poorer performance in exchange for a lower value of ρ. The small improvement at moderate values of λ, though, is not too surprising. If the direction K corresponded to one that is especially noisy, reducing the importance of that direction can lead to improved performance. We will see more on this in Section 6.5.

The feature-level and feature+score combination methods perform approximately equal at around 1% EER, while, for $\lambda = 0$, the score-level combination has a significantly worse performance of 1.58%. Nevertheless, as λ grows, the performance of the score-level combination using the anticorrelation kernel improves significantly (from 1.58% to 0.56% when λ goes from 0 to 10^4), making it the best-performing system. Overall, we see a reduction in EER of around 50%, relative to the EER of the best combined system when the anticorrelation kernel is not used.

Figure 6 shows the scatter plot of scores (on the training data) for both systems corresponding to $\lambda = 0$ and $\lambda = 10000$. We can see that for the large value of λ, the within-class covariances have been largely reduced. We can also see that the separation of the two classes is better for the larger λ, which explains the performance improvement observed in Figure 5.

Figure 7 shows the results for the score-level combination of systems B and S with $\lambda = 0$ (that is, without using the proposed method), the score-level combination with $\lambda = \infty$, the feature-level combination, and the feature+score combination, for several settings of the simulation parameters N, m, and d. For the score-level combination with $\lambda = \infty$ we also present the results obtained when computing K using our knowledge of the model that generated the data. Since we are creating

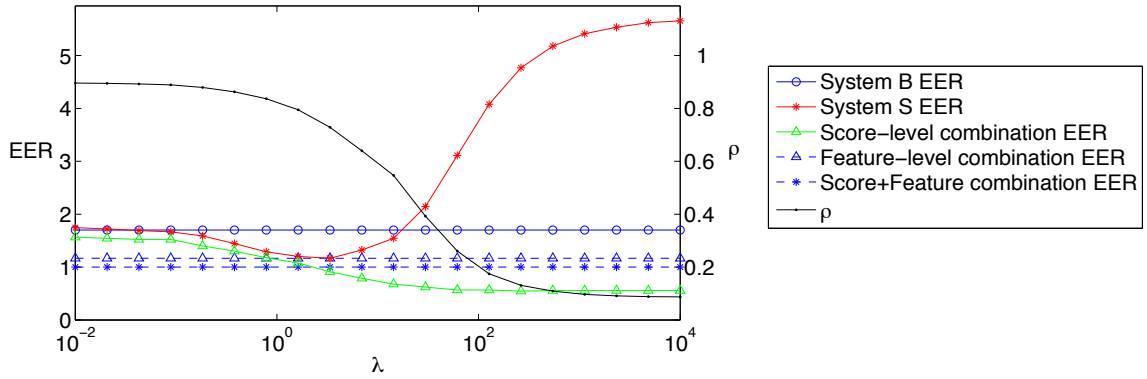


Figure 5: Error of individual systems and their combination, and value of the ρ coefficient as a function of λ for an artificial problem. The EER of the combination is reduced from 1.58% to 0.56% as λ increases.

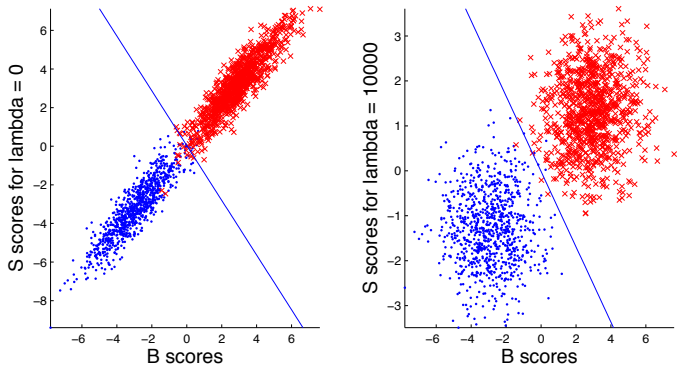


Figure 6: Scores from system B versus scores from system S for two values of λ.

the data ourselves according to a model, we can compute K exactly using (11) instead of (12). It can be shown that, for our setup, $K = \frac{1}{2}\sum_{y=\{a,b\}}(C_{12}C_{11}^t + C_{22}C_{12}^t)w_B$, where w_B is the SVM weight vector for system B and C_{ij} is block ij of size $d/2 \times d/2$ of matrix C. For each set of parameters, 10 different random seeds were used to generate the data, keeping matrix C equal for all 10 experiments. Each bar shows the first quartile, the median, and the third quartile of the set of EERs obtained from the 10 simulations.

The figure shows that feature+score combination is significantly better than score-level or feature-level combinations only when the task is easier ($m = 3.0$) and many samples are available for training ($N = 900$). Feature-level combination is optimal when the task is harder ($m = 1.8$) and the number of features is small ($d = 250$) particularly when enough samples are available for training ($N \geq 300$). Plain score-level combination (without anticorrelation) performs comparably to feature-level combination when the number of training samples is small and the number of features is large, in which case the feature-level combination suffers from the additional complexity. In all cases, though, the anticorrelation method (last two bars) is either comparable or significantly better than

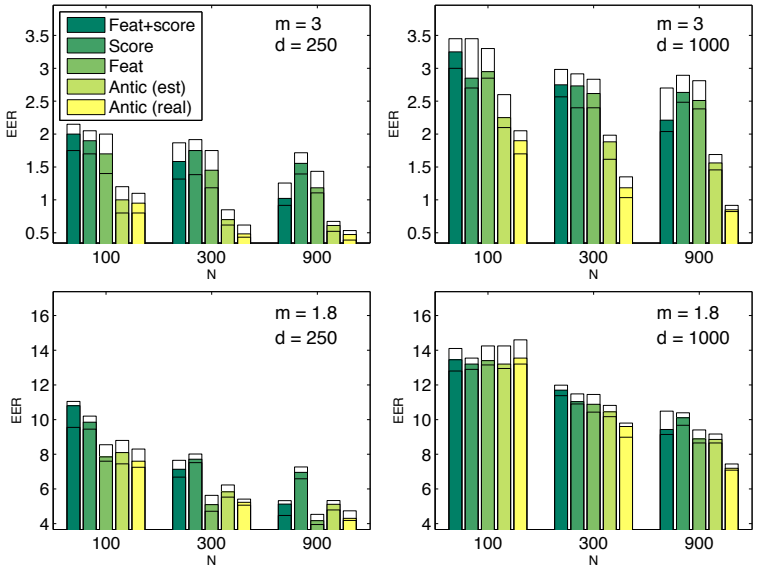


Figure 7: Comparison of EER on simulated data for the score-level combination of system B with S with $\lambda = 0$ (called *Score* in the legend), the score-level combination with $\lambda = \infty$ (*Antic*) using the estimated value for K (*est*) or the value obtained from the model (*real*), the feature-level combination (*Feat*), and the feature+score combination (*Feat+score*) for several values of the simulation parameters. For each pair of m (distance between means) and d (feature vector dimension), three values of N (number of samples from class b) are explored.

plain score-level, feature+score and feature-level combinations. For $m = 3.0$, the anticorrelation method significantly outperforms all other combination methods for both values of d and the three values of N. Gains are smaller or disappear when the task becomes harder ($m = 1.8$).

The difference between the fourth and the fifth bars in each set is due only to the difference in the K vector used. The K is estimated using the data (Equation 12) for the fourth bar and using the model (Equation 11) for the fifth bar. We can see that when the dimension of the feature vector Z_2 is large, the difference between the fourth and the fifth bars becomes larger, indicating that in these situations the estimation of K is noisier. This is also evident from looking at the reduction in ρ achieved for the different values of d when using the data to estimate K. For $d = 250$, the reduction is around 90%, while for $d = 1000$ the reduction is only around 60%. In most experiments with $d = 1000$ the value of ρ when using the estimated value of K does not go under 0.30. This means that for higher-dimensional vectors, the estimation of K is harder than for lower-dimensional ones. Nevertheless, even in those cases, the combination using the anticorrelated system with the estimated K is, in most cases, better than the original score-level combination.

6. Experiments on Speaker Verification

Speaker verification is the task of deciding whether or not a speech sample was produced by a certain target speaker. It is a binary classification task where the two classes are *true-speaker* and *impostor*. To test the proposed method we use a standard UBM-GMM system, a cepstral supervector SVM system, an MLLR-based system, and a prosodic system. We show results using the proposed kernel on all possible combinations involving two systems (two-way combinations) and a variety of combinations involving three and four systems (three-way and four-way combinations).

6.1 Databases and Error Measures

Experiments were conducted using data from the NIST speaker recognition evaluations (SRE) from 2005 and 2006. Each speaker verification trial consists of a test sample and a speaker model. The samples are one side of a telephone conversation with approximately 2.5 minutes of speech. We consider the 1-side training conditions in which we are given 1 conversation side to train the speaker model. This conversation corresponds to a positive example when training the SVM model for the speaker. The data used as negative examples for the SVM training and to estimate the K vectors is taken from 2003 and 2004 NIST evaluations along with some FISHER data, resulting in a total of 4355 samples. The tasks contain 26,270 trials for SRE05, and 21,343 for SRE06. In both cases around 1/10th of the trials are target trials. Trials are created by reusing the conversations from a few hundred speakers as train and test samples, sometimes as target speakers, sometimes as impostors. A total of 598 distinct models for SRE05 and 584 for SRE06 are created, some of them corresponding to different conversations from the same speaker.

The performance measures used in this section are the EER and NIST's detection cost function (DCF). The DCF is defined as the Bayesian risk with probability of target equal to 0.01, cost of false alarm equal to 1, and cost of miss equal to 10. The DCF is affected both by the discrimination power of the system and its calibration, given by the choice of threshold that is believed to minimize it (Brummer and du Preez, 2006). In this paper, we will not explore the calibration issue, which is, in itself, a large field of study in the biometrics community. We will present results in terms of the DCF achieved when choosing the threshold that minimizes it on the test data. This measure is commonly called *minimum* DCF and it measures *how much information the detector could have delivered to the user, if the calibration had been perfect* (Brummer and du Preez, 2006).

The EER and the DCF are two points in the receiver operating characteristic (ROC) curve of a system and they give a more complete picture of the behavior of the system for different operating points than the EER alone. Even though the theory in Section 3 was developed for EER, we will see that improvements are obtained for both performance measures.

6.2 Individual System Descriptions

The systems chosen to run the experiments in this paper are representative of the systems being used in most state-of-the-art speaker recognition systems. Somewhat simplified versions of the best-performing systems were used, in order to facilitate the large amount of computationally costly experiments that were run. A brief description for each of the systems follows.

6.2.1 UBM-GMM SYSTEM (G)

This is probably the most widely used paradigm for speaker verification. A Gaussian mixture model (GMM) is trained using data from many different speakers and recording conditions to create a *universal* background model (UBM). The target speaker models are trained by maximum a posteriori adaptation of the background model to the training data. For a given test sample the logarithm of the ratio of the likelihoods for the target model and the background model is used as a score. The system used here is based on 13 mel frequency cepstral coefficients (MFCCs) without the zeroth-order coefficient, and first-, second-, and third-order difference features, resulting in 52-dimensional feature vectors. The features are modeled by 2048 mixture component GMMs. Only the GMM means are adapted to the observed data, leaving variances and weights untouched. For implementation details on this system see Shriberg et al. (2005).

6.2.2 SUPERVECTOR-SVM SYSTEM (V)

This system (Campbell et al., 2006) is a variation of the UBM-GMM system, where SVMs are used to obtain scores. For each sample, the means of the UBM-GMM are adapted to the sample's data and stacked together in a single high-dimensional feature vector. A set of held-out samples (generally the same samples used to create the UBM-GMM) is used as negative examples when training the SVM, while the target sample is used as the positive example. These features are used to train a model using support vector regression with an inner-product kernel. The signed distance to the hyperplane is then used as the output of the system. For this system we use a 512-component background model. Since the dimension of the original space is 52, the final dimension of the feature vectors is given by $512 \times 52 = 26,624$. Larger background models have been found to give slightly better performance but increase the computational cost of the experiments. We found 512 components to give a good balance between performance and computational cost of the system.

6.2.3 MLLR-SVM SYSTEM (M)

The MLLR-SVM system (Stolcke et al., 2007, 2006) uses the speaker adaptation transforms used in the speech recognition system as features for speaker verification. A total of four affine 39x40 transforms is used to map the Gaussian mean vectors from speaker-independent to speaker-dependent speech models; two transforms each are estimated relative to male and female recognition models, respectively. The transforms are estimated using maximum-likelihood linear regression (MLLR) and can be viewed as a text-invariant encapsulation of the speaker's acoustic properties. The transform coefficients form a 6,240-dimensional feature space. Each feature dimension is rank normalized by replacing the value with its rank in the background data, and scaling ranks to lie in the interval $[0, 1]$. The resulting normalized feature vectors are then modeled using the same procedure as for the supervector-SVM system. The system described in this paragraph is a simplified version of our best performing MLLR-SVM system which uses a total of 16 transforms and has approximately 25% lower EER than the 4-transform system (Stolcke et al., 2007). Initial experiments (not shown here) indicate that improvements from the anticorrelation method are still obtained when the more complex MLLR system is used, with similar relative gains as the ones shown here.

6.2.4 SNERF-SVM SYSTEM (S)

This system models syllable-based prosodic NERFs (nonuniform extraction region features) (Shriberg et al., 2005). Features are based on estimated F_0, energy, and duration information extracted over syllables inferred via automatic syllabification based on automatic speech recognition output. Prosodic feature sequences are transformed into fixed-length vectors by a particular implementation of the Fisher score (Ferrer et al., 2007). In this paper, only features modeling sequences of two syllables are used. In previous work we have found that these features by themselves yield a performance almost as good as using features extracted for sequences of 1, 2, and 3 syllables together. The resulting feature vector, of dimension 13,343, is first rank-normalized (as in the MLLR system) and modeled using the same procedure as for the supervector-SVM system.

6.3 Application of the Proposed Method to the Speaker Verification Problem

Most speaker verification systems that use SVMs as models consider each train or test utterance as a single sample. If necessary, as in the case of the SNERF features and many other cases presented in the literature (Ferrer et al., 2006; Brummer et al., 2007; Reynolds et al., 2005), a transform is applied to the input features prior to SVM modeling in order to convert them into a single fixed-length vector. In other cases, such as the MLLR system, the features are directly generated as a single fixed-length vector. In our experiments, since we are presenting results on the 1-side training condition from NIST evaluations, this implies that only one positive sample is available during training for each speaker model. This means that the estimation of K in (12) will be given only by impostor samples. These impostor samples are extracted from a held-out set. For each target model in the task definition we require a separate vector K. This results in significant overhead during training since each model from system B must be tested against the held-out set used to compute K. Nevertheless, this has no effect at test time. Once the vector K for each target model is computed, obtaining the score for a new test is almost as fast as for a linear kernel SVM.

6.4 Results

Table 1 shows the results on SRE05 and SRE06 data for the individual systems. Each system is represented by a single letter: **G** for the GMM-UBM, **V** for the Supervector-SVM, **M** for the MLLR-SVM, and **S** for the SNERF-SVM. For the SVM systems (Supervector, MLLR, and SNERFs), we show the baseline results (training the SVM with an inner product kernel) and the results obtained by training the target SVMs using the kernel in (14) with K computed using the scores corresponding to each of the other three systems. This is indicated by the use of a subindex corresponding to the system with respect to which the anticorrelation is performed. For example, $\mathrm{M_G}$ corresponds to a system that uses the MLLR features and anticorrelation kernel with respect to the GMM-UBM system (that is, with K given by the vector of average class-conditional covariances between the MLLR features and the scores from the GMM-UBM system). A list of subindices corresponds to performing anticorrelation with respect to more than one system as described in Section 4.4. Hence, system $\mathrm{S_{M,V_M}}$ corresponds to system S anticorrelated with respect to systems M and $\mathrm{V_M}$. In all cases the anticorrelation results shown correspond to $\lambda = \infty$, which implies that the resulting weight vector will not have a component in the direction of K. This was shown to be optimal in the simulated experiments and in several preliminary experiments with the systems from this table.

System	SRE05		SRE06	
	DCF	EER%	DCF	EER%
G	0.303	7.259	0.306	5.639
M	0.266	7.096	0.221	4.979
M_G	0.297	8.564	0.243	5.879
M_V	0.289	8.768	0.254	6.119
M_S	0.271	7.586	0.229	4.979
V	0.205	5.465	0.174	3.419
V_G	0.223	6.525	0.179	4.019
V_M	0.196	5.465	0.168	3.179
V_S	0.203	5.506	0.168	3.359
S	0.545	14.233	0.548	12.777
S_G	0.560	15.008	0.562	13.077
S_M	0.577	15.824	0.579	13.497
S_V	0.569	15.253	0.573	13.137
S_{M,V_M}	0.588	16.150	0.592	14.157

Table 1: Results for individual systems (G: GMM-UBM, V: Supervector-SVM, M: MLLR-SVM, S: SNERF-SVM) with inner product kernel and anticorrelation kernel. When the anticorrelation kernel is used, a subindex indicates the name of the system or systems with respect to which the anticorrelation is performed.

It can be seen that in most cases, using the anticorrelation kernel results in a degradation in performance in the system. A notable exception is the result for system V_M (Supervector features using anticorrelation kernel with respect to the MLLR-SVM system). In this case, preventing the use of the direction given by K results in a significant gain in performance. This could happen if vector K corresponded to some noisy direction that, when ignored, allowed for other more robust directions to be used. This effect was also observed in the simulations and will be discussed in more detail in Section 6.5.

Table 2 shows results for all the possible two-way combinations of the four individual systems. For the score-level combinations (indicated with "+") we used a linear model trained on SRE05 data using logistic regression. Feature-level and feature+score combinations are indicated with "$\cup$". The symbol indicates feature concatenation. In the case of feature-level combination, features from both systems are directly concatenated. In the case of feature+score combination, the features from one system are concatenated with the scores from the other (this is indicated with the subscript "scores"). Whenever feature concatenation is performed, a weight is used (as described in Section 4.6) to emphasize the features from one set versus the other. This weight is tuned on SRE05. Since tuning this weight is extremely demanding computationally (it requires running a full classification experiment for each value of the weight), only feature+score combination experiments involving the MLLR systems are run. All possible feature-level combinations are shown, since there are only three of them.

We can see that every time a score-level combination (+) is done between systems X and Y_X,[6] the performance is better than that for the combination of X and Y (with the single exception of SRE05's EER for $V + S_V$). That is, applying the anticorrelation kernel to system *Y* always gives a gain in the combination performance, even though in most cases system Y_X has worse individual performance than system Y. Feature-level combinations, $X \cup Y$, do not show any advantage over the much simpler score-level combination $X + Y$. This simply indicates that the increased model complexity of the feature-level combination cannot be properly handled with the available amount of training samples. Similarly, the feature+score combinations, $X \cup Y_{\text{scores}}$, do not give any consistent improvement over the score-level combinations $X + Y$.

The most notable gain from using the anticorrelation kernel is found for the combination $M + V_M$. The relative gain in EER with respect to system $M + V$ is 16%. As mentioned above, system V_M is in fact better performing than system V. That is, anticorrelating with respect to M does not degrade its performance but improves it. This fact, together with a reduced impostor correlation between the systems, explains the observed large gain.

The last column in Table 2 shows the class-conditional correlation between the two systems being combined for the impostor and the target samples in SRE06 data. As we can see, the impostor correlation is significantly reduced when the proposed method is used, even though it does not reach a zero value. (This problem was also observed in the simulated experiments for large values of the feature vector dimension d.) This could mean that the amount of data used for the computation of K (4355 samples) is not enough to obtain a robust estimation of the statistics in the test data or that the statistics in the test data are not the same as those in the held-out set used to compute K. Furthermore, we can see that the target correlation remains almost unchanged by the application of the anticorrelation kernel. This is reasonable, since the vector K is computed without the use of any target data. The fact that the target correlation is not reduced when K is computed only over impostor samples suggests that the correlations in both populations are not equal and one cannot be predicted from the other. Nevertheless, a reduction in either of the class-conditional correlation coefficients can result in a reduction of ρ as given by (9).

Finally, Table 3 shows some three-way and four-way combination results. The first four results correspond to the combination of the three SVM systems. The three combinations shown that use the anticorrelation kernel on the V and S systems perform very similarly, resulting in a performance improvement of 18.9% on the SRE06 EER with respect to the baseline combination. We can see that using multiple anticorrelation on the S system with respect to the other two systems already in the combination (M and V_M) does not lead to further improvements (line $M + V_M + S_{M,V_M}$). This is, in fact, good news, since doing the multiple anticorrelation involves a significant amount of extra computation to obtain the K vector of system S with respect to the scores from V_M.

The last three lines in Table 3 show the results on some four-way score-level combinations. We see a large improvement of 19.2% when a successive anticorrelation procedure is used, where each new model is anticorrelated with the one previously added to the combination. The best three- and four-way combinations all include the V_M system. This was expected since the two-way gain from using anticorrelation on that system was the largest among all two-way combinations.

An overall observation from this table is that the proposed method performs better on SRE06 data than on SRE05 data, even though the combiner is trained on SRE05 data, making the SRE05 results slightly optimistic. We believe that this might be a consequence of a better statistical match

6. Letters X and Y are used in this section to indicate any two systems. Hence, $X, Y \in \{G, M, V, S\}$.

System	SRE05		SRE06		
	DCF	EER%	DCF	EER%	CorI / CorT
$G_{scores} \cup M$	0.219	5.710	0.189	4.079	-
$G + M$	0.226	5.750	0.201	4.019	0.51/0.79
$G + M_G$	**0.210**	**5.465**	**0.188**	**3.779**	0.26/0.76
$G + V$	0.203	5.383	0.191	3.419	0.74/0.88
$G + V_G$	**0.194**	**5.261**	**0.182**	**3.239**	0.35/0.86
$G + S$	0.219	5.587	0.232	4.499	0.16/0.45
$G + S_G$	**0.216**	**5.465**	**0.224**	**4.259**	0.11/0.44
$M \cup V$	0.188	5.383	0.164	3.239	-
$M \cup V_{scores}$	0.176	5.098	0.148	3.119	-
$M + V$	0.180	5.139	0.160	3.299	0.58/0.87
$M + V_M$	**0.161**	**4.812**	**0.140**	**2.759**	0.37/0.84
$M_V + V$	0.171	4.976	0.142	3.179	0.31/0.83
$M \cup S$	0.245	6.403	0.200	4.379	-
$M \cup S_{scores}$	0.225	**5.913**	**0.190**	4.439	-
$M + S$	0.224	6.158	0.194	4.319	0.22/0.55
$M + S_M$	0.221	5.995	0.191	**4.079**	0.14/0.52
$M_S + S$	**0.215**	5.995	0.191	**4.079**	0.15/0.53
$V \cup S$	0.183	5.057	0.152	3.179	-
$V + S$	0.163	4.609	0.146	3.239	0.19/0.50
$V + S_V$	0.161	4.812	0.145	3.119	0.13/0.49
$V_S + S$	**0.159**	**4.527**	**0.137**	**2.999**	0.15/0.49

Table 2: Results for two-way combinations of the systems in Table 1. Symbol "$\cup$" indicates feature concatenation. Hence, $M \cup S$ corresponds to feature-level combination, while $M \cup S_{scores}$ corresponds to feature+score combination of systems M and S. Symbol "$+$" indicates score-level combination. The last column shows the correlations between the pair of systems being combined, for the impostor (CorI) and the target (CorT) samples.

between SRE04 data (used to compute the *K* vectors) and SRE06 data than between SRE04 and SRE05 data.

Evidently, the behavior found in these experiments cannot be expected to generalize to all possible sets of features and tasks. For example, if much smaller sets of features were used and enough training data was available for both classes, feature-level combination might result in better performance than score-level combination (as seen in the simulated experiments). Nevertheless, the systems used here are representative of the kinds of systems used for speaker recognition on state-of-the-art systems, where very large feature vectors have been found to outperform smaller ones. Furthermore, the small amount of positive training examples is an inherent characteristic of the speaker recognition task. Finally, these characteristics are found in many other modern machine learning tasks, a notable example being classification of microarray expression data, where the

System	SRE05		SRE06	
	DCF	EER%	DCF	EER%
$M+V+S$	0.149	4.690	0.134	3.179
$M+V_M+S_M$	0.133	**4.364**	0.117	**2.579**
$M+V_M+S_V$	**0.132**	4.445	**0.115**	**2.579**
$M+V_M+S_{M,V_M}$	0.133	4.405	0.116	**2.579**
$G+M+V+S$	0.149	4.649	0.141	3.119
$G+M_G+V_G+S_G$	0.147	4.323	0.132	2.700
$G+M_G+V_M+S_V$	**0.137**	**4.160**	**0.120**	**2.519**

Table 3: Results for three-way and four-way combinations of the systems in Table 1.

number of features is in tens of thousands, and the number of samples for most studies is limited to tens or at most hundreds.

6.5 Interaction with Intersession Variability Compensation

The variability found across different recordings of the same speaker is commonly called *intersession variability* (ISV). This effect can be caused by a mismatch in channel conditions, emotional state, phonetic content, and so on, and it is one of the biggest sources of errors in speaker verification. Several methods have been developed to reduce the effect of intersession variability.

In the realm of SVMs, the most widely used ISV compensation (ISVC) method is nuisance attribute projection (NAP) (Solomonoff et al., 2004; Campbell, 2006). NAP consists of estimating the directions in the feature space that vary with the sessions and then projecting the samples on the complement of the space determined by those directions. The *noisy* directions are calculated as the first few eigenvectors of the within-speaker covariance matrix. This matrix is in turn estimated from held-out data for which several samples of each speaker are available. All speakers are pulled together and a single within-speaker covariance matrix is estimated. The number of directions to be eliminated from the feature vectors is determined empirically. Both NAP and the anticorrelation method presented here transform the features by eliminating certain directions. In the case of NAP these directions are the ones estimated to have information superfluous to the task of speaker verification. In the anticorrelation procedure, a single direction is eliminated: the one that maximizes the average class-conditional covariance between the two systems being combined.

In the case of UBM-GMM systems and systems like the supervector-SVM, which are based on the UBM-GMM models, a different type of ISVC based on the factor analysis method can be applied (Kenny and Dumouchel, 2004; Kenny et al., 2007). The method is based on the assumption that a supervector m corresponding to a certain sample can be decomposed into a speaker-dependent and a channel-dependent component. That is, $m = s + c$, where s is a speaker supervector and c a channel supervector. Furthermore, c is assumed to be given by ux, where u is a low-rank matrix and x is a normally distributed random vector. The components of vector x are called the *channel factors* and the columns of matrix u the *eigenchannels*. In order to estimate the matrix u, a database with several samples for each speaker (as the one required for NAP) is needed. In some models, s is further decomposed into different terms. Many different methods have been used to estimate and

System	SRE05		SRE06		SRE06
	DCF	EER%	DCF	EER %	CorI / CorT
M	0.230	6.525	0.195	4.139	-
M_V	0.257	7.667	0.217	4.919	-
V	0.171	4.935	0.145	2.819	-
V_M	0.177	5.302	0.144	2.879	-
$M+V$	0.144	4.568	0.120	2.639	0.47/0.87
$M+V_M$	0.144	4.690	0.117	2.579	0.40/0.86
M_V+V	0.142	4.649	0.119	2.639	0.36/0.85

Table 4: Results for individual systems and their combinations after ISVC.

compensate for the channel factors. The method used in this paper for the supervector system is described by Matrouf et al. (2007).

Table 4 shows a subset of results for two of the systems described in Section 6.2 when ISVC is applied. Factor analysis is used for the Supervector-SVM system, and NAP is used for the MLLR-SVM system. We can see that, for these systems, the anticorrelation kernel does not reduce the class-dependent correlations between pairs of systems enough to result in a gain in the combination performance. The fact that applying the anticorrelation kernel does not result in a significant reduction of the class-dependent correlations between the systems indicates that the K vectors computed from the held-out data are not a good estimation of the K vectors in the test data. If we compare the impostor correlation between the same pair of systems when no ISVC is applied (Table 2) versus the impostor correlation when ISVC is applied (Table 4) we see that ISVC'ed systems are much less correlated. Furthermore, the correlation when the anticorrelation method is applied results in similar values for ISVC'ed and non-ISVC'ed systems. This suggests that much of the correlation between the non-ISVC'ed systems is due to intersession variability effects. This intuition is confirmed by computing the projection of the K direction on the NAP directions, which shows that K is mostly in the direction of the first few NAP directions (when they are sorted by the size of the corresponding eigenvalue). Hence, when ISVC is applied to a system, the ISV effects are eliminated (or reduced), resulting on less-correlated systems that combine better with each other. For example, the performance for combination $M+V$ for non-ISVC'ed systems (3.299% from Table 2) is only 3.5% better than the best of the two individual performances (3.419% from Table 1), while the performance for that same combination but using ISVC'ed systems (2.639% from Table 4) is 6.4% better than the best of the two individual performances (2.819% from Table 4).

These observations explain the reduction in class-conditional correlation between pairs of systems when ISVC is applied to them, since a large part of the class-conditional correlation is due to the intersession variability. On the other hand, it does not explain why the correlation cannot be further reduced after the intersession variability noise has been eliminated. The reason for this is simply that the vector K estimated for each target model does not predict the direction of maximum impostor covariance on the test data. On the training data, we know that the covariance for the impostor cloud is necessarily pushed to zero when $\lambda = \infty$, but we do not observe the same behavior on the test data. This is because, as we saw in the simulations, the estimation of the direction K

gets noisier for larger feature vectors. Hence, only a very noticeable effect (like the intersession variability one) can be robustly estimated.

We believe that the observations presented in this section do not invalidate the usefulness of the method for the speaker verification task. As mentioned in Kenny et al. (2007), for ISVC to work, a well-balanced database is required where samples from several different recording conditions for each speaker are available. This kind of database is not easy to obtain. When such a database is not available, ISVC cannot be applied to the systems. In these cases, the anticorrelation method proposed here would be able to bring back some of the gain that ISVC would result in if the right database was available.

7. Conclusions

While speaker verification systems have seen large gains in performance from *ad hoc* combination of several component systems, a unified framework for joint development of a combined system that ensures system diversity has been lacking. The component systems are trained in isolation to maximize individual performance rather than the overall system being trained to maximize combined performance. In this work, we presented a simple model for the system combination problem and found the performance of the combined system to be a function of the performance of the individual systems being combined and a "correlation coefficient" obtained from the average class-condition covariance of the vector of scores. Based on this result we presented a technique for taking into account the characteristics of the scores from a set of fixed existing systems during the development of a new SVM system in order to improve the combined system performance. This is realized through a modification of the SVM optimization problem via the introduction of a regularization term involving the covariance between the scores of the previously existing systems and the input features to the SVM, explicitly encouraging diversity of the resulting system ensemble. The trade-off between the individual performance of the SVM system and the inter-system average class-conditional covariance is reflected in the optimization through the introduction of the Lagrange multiplier λ. The technique can be implemented cheaply through the use of a simple kernel function, which we call *anticorrelation* kernel.

We show the effectiveness of the anticorrelation technique in a series of simulated experiments and in speaker verification experiments on the 2005 and 2006 NIST SRE tasks using four component systems: a standard UBM-GMM system, a cepstral supervector system, an MLLR-based system, and a prosodic system. We show results using the proposed kernel on all possible combinations involving two systems (two-way combinations) and on some combinations involving three and four systems. We demonstrate a performance gain of around 19% for a four-way combination using the anticorrelation kernel with respect to the performance of the combination obtained without anticorrelation. When the same four speaker verification systems are compensated for intersession variability, the gains from the anticorrelation method disappear. Our analysis indicates that the reason for this is that much of the correlation between the systems is, in fact, due to the intersession variability. Once systems are compensated for this variability, the remaining correlation is too hard to estimate robustly. The anticorrelation method can then be seen as a replacement for intersession variability compensation methods when the right databases are not available for the estimation of the matrices needed for those methods.

The fact that, in our experiments, the combination performance improves monotonically as λ grows and ρ decreases indicates that the optimal trade-off between the performance of the individual

system and the value of ρ probably occurs at a negative value of ρ. Since we are using $\lambda\sigma_{SB}^2$ as our regularization term, σ_{SB} and, with it, ρ are forced toward zero and negative values will be unlikely to occur in our setup. Using a linear term $\lambda\rho$ in the objective function would allow ρ to become negative, perhaps leading to better combination results. Solving this optimization problem, though, requires the use of general-purpose convex optimization software, which would be too slow for our purposes, or the development of a solver specifically designed for it. This is a direction we plan to explore.

We note that the anticorrelation technique is general in that it can be applied to any binary classification task for which more than one system can be trained and at least one of them is an SVM. Many modern machine learning problems have these characteristics, among them microarray gene expression classification problems (Brown et al., 2000), biometric tasks (Roy and Bhattacharya, 2005; Heisele et al., 2003), and a variety of other classification tasks (Sebastiani, 2002). The proposed method has the potential to lead to significant gains on some of those tasks and many others depending on the nature of the features used, their dimension, the number of samples available for training, the absolute performance of the systems, and so on. Finally, since the implementation of the proposed method simply reduces to the use of a specific kernel function, any statistical procedure that can be kernelized (of which SVMs are simply one example) could potentially benefit from it.

Acknowledgments

We thank Robert M. Gray and Robert Tibshirani for helpful comments and discussions. We also thank our colleagues at SRI's Speech Technology and Research Laboratory. In particular, we thank Sachin Kajarekar, Andreas Stolcke, and Nicolas Scheffer for providing some of the features and systems used in these experiments. We are also grateful to the anonymous reviewers who provided valuable feedback that greatly improved the paper. This research was funded by NSF CNS-0652510 at Stanford University and through a development contract with Sandia National Laboratories by the National Geospatial-Intelligence Agency (NGA) under National Technology Alliance (NTA) Agreement Number NMA 401-02-9-2001 at SRI International. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the NSF, NGA, the United States Government, or Rosettex.

Appendix A. Proof of Upper Bound on EER (Equation 4)

We wish to prove that if a set of scores F is distributed such that $F \mid Y = y \sim \mathcal{N}(\mu_y, \sigma_y^2)$, for $y = a, b$ (that is, the class-conditional distributions are Gaussian), the EER obtained when the class is estimated as in (1) is upper bounded by $\phi(-\frac{1}{2}\frac{\delta\mu}{\sigma})$, where $\delta\mu = \mu_b - \mu_a$, and σ satisfies $\sigma \geq (\sigma_a + \sigma_b)/2$.

For the case in which $\sigma_a = \sigma_b = \sigma$, the threshold t^* corresponding to EER is given by $t_s^* = (\mu_a + \mu_b)/2$ (s here stands for *same* variance) since this is the value that results in the rate of false acceptances ($e_{b|a}$) being equal to the rate of false rejections ($e_{a|b}$). Replacing this threshold in (2) or (3), we get that $\mathrm{EER}_s = \phi(-\frac{1}{2}\frac{\delta\mu}{\sigma})$. The upper plot in Figure 8 illustrates this case.

When variances are not equal, the EER threshold is no longer at half the distance between the two means. Nevertheless, we can find the approximate location of the threshold by mapping the value of t_s^* to two new locations, t_{db} and t_{da} (d stands for *different* variance) such that in one case the rate of false rejections is equal to EER_s and in the other case the rate of false acceptances is equal to EER_s. The EER threshold for the unequal variance case, t_d^*, will then be somewhere between these two points, since being to the left or to the right of both of them would result in the rates of false rejections and false acceptances being different from each other.

The values t_{da} and t_{db} are determined such that $e_{b|a}(t_{da}) = \text{EER}_s$ and $e_{a|b}(t_{db}) = \text{EER}_s$, respectively (with $e_{a|b}$ and $e_{b|a}$ defined in (2) and (3)), and they are given by

$$\begin{aligned} t_{db} &= \sigma_b/\sigma(t_s^* - \mu_b) + \mu_b \\ t_{da} &= \sigma_a/\sigma(t_s^* - \mu_a) + \mu_a. \end{aligned}$$

If $t_{da} \leq t_{db}$ then there will be a threshold t_d^* between t_{da} and t_{db} for which $e_{a|b}(t_d^*) = e_{b|a}(t_d^*) = \text{EER}_d \leq \text{EER}_s$. This is the case illustrated in Figure 8. So, if we can find some σ such that $t_{da} \leq t_{db}$, we can replace this value in $\text{EER}_s = \phi(-\frac{1}{2}\frac{\delta\mu}{\sigma})$ to get the desired upper bound. Now,

$$t_{da} - t_{db} = \frac{\sigma_a}{\sigma}(t_s^* - \mu_a) + \mu_a - \frac{\sigma_b}{\sigma}(t_s^* - \mu_b) - \mu_b = \delta\mu\left(\frac{\sigma_b + \sigma_a}{2\sigma} - 1\right).$$

Since we are assuming that $\delta\mu > 0$, we see that $t_{da} \leq t_{db}$ if and only if $\sigma \geq (\sigma_a + \sigma_b)/2$. When $\sigma = (\sigma_a + \sigma_b)/2$ we get $t_{da} = t_{db}$, which implies that $\text{EER}_d = \text{EER}_s$. Hence, we can always compute EER_d as $\phi(-\frac{\delta\mu}{\sigma_a+\sigma_b})$. It is easy to prove that $\sigma = \sqrt{(\sigma_a^2 + \sigma_b^2)/2}$ (the square root of the average variance instead of the average of the standard deviations) also satisfies $\sigma \geq (\sigma_a + \sigma_b)/2$.

References

T. Anderson and R. Bahadur. Classification into two multivariate normal distributions with different covariance matrices. *The Annals of Mathematical Statistics*, 33(2):420–431, June 1962.

L. Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

G. Brown, J. Wyatt, R. Harris, and X. Yao. Diversity creation methods: A survey and categorisation. *Journal of Information Fusion*, 6:5–20, 2005a.

G. Brown, J. L. Wyatt, and P. Tiňo. Managing diversity in regression ensembles. *Journal of Machine Learning Research*, 6:1621–1650, 2005b.

M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. Furey, M. Ares Jr., and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. In *Proceedings of the National Academy of Sciences*, volume 97, pages 262–267, 2000.

N. Brummer, L. Burget, J. Cernocky, O. Glembek, F. Grezl, M. Karafiat, D. van Leeuwen, P. Matejka, P. Schwarz, and A. Strasheim. Fusion of heterogeneous speaker recognition systems in the STBU submission for the NIST speaker recognition evaluation 2006. *IEEE Transactions on Audio, Speech and Language Processing*, 15(7):2072–2084, September 2007.

N. Brummer and J. du Preez. Application independent evaluation of speaker detection. *Computer Speech and Language*, 2006.

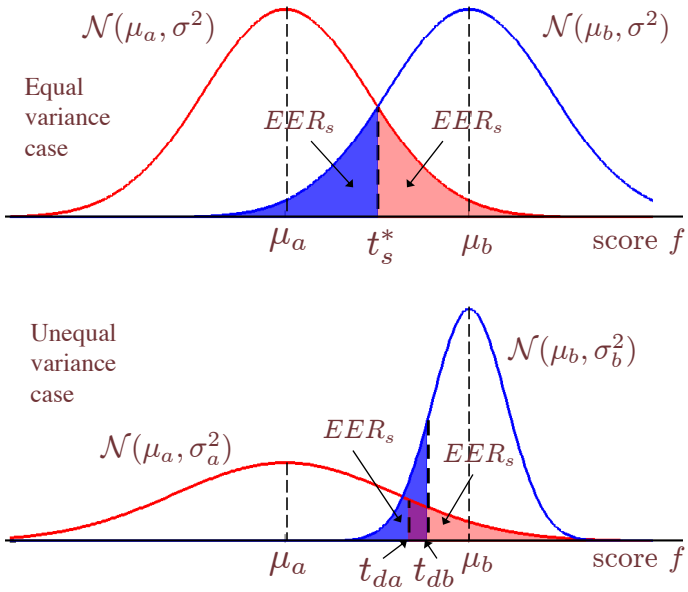


Figure 8: Proof of (4). The upper figure illustrates the computation of EER_s, the EER for the equal variance case. In the lower figure, the unequal variance case is considered. The x-axis values t_{da} and t_{db} are computed such that the area of the two shaded regions coincides with the area of the respective shaded regions in the upper plot. All four shaded areas are then equal to EER_s. The figure illustrates the case in which $t_{da} \leq t_{db}$. In this case, there will be a threshold t_d^* somewhere between t_{da} and t_{db} which corresponds to EER_d, the EER for the unequal variance distributions. From this, we conclude that, if $t_{da} \leq t_{db}$, then $\text{EER}_d \leq \text{EER}_s$.

W. Campbell. Compensating for mismatch in high-level speaker recognition. In *Proceedings of the Speaker and Language Recognition Workshop, Odyssey 2006*, Puerto Rico, USA, June 2006.

W. Campbell, D. Sturim, and D. Reynolds. Support vector machines using GMM supervectors for speaker verification. *IEEE Signal Processing Letters*, 13(5):308–311, May 2006.

N. Dehak, P. Kenny, and P. Dumouchel. Continuous prosodic features and formant modeling with joint factor analysis for speaker verification. In *Proceedings of the 10th European Conference on Speech Communication and Technology (Interspeech 07)*, Antwerp, August 2007.

L. Ferrer, M. Graciarena, A. Zymnis, and E. Shriberg. System combination using auxiliary information for speaker verification. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Las Vegas, April 2008a.

L. Ferrer, E. Shriberg, S. Kajarekar, and K. Sönmez. Parameterization of prosodic feature distributions for SVM modeling in speaker recognition. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Honolulu, April 2007.

L. Ferrer, E. Shriberg, S. Kajarekar, A. Stolcke, K. Sönmez, A. Venkataraman, and H. Bratt. The contribution of cepstral and stylistic features to SRI's 2005 NIST speaker recognition evaluation system. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 101–104, Toulouse, May 2006.

L. Ferrer, Kemal Sönmez, and E. Shriberg. An anticorrelation kernel for improved system combination in speaker verification. In *Proceedings of the Speaker and Language Recognition Workshop, Odyssey 2008*, Stellenbosch, South Africa, January 2008b.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

G. Fumera and F. Roli. A theoretical and experimental analysis of linear combiners for multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

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

B. Heisele, P. Ho, J. Wu, and T. Poggio. Face recognition: Component-based versus global approaches. In *Computer Vision and Image Understanding*, volume 91(1), pages 6–12. Elsevier, 2003.

F. Huenupán, N. B. Yoma, C. Molina, and C. Garretón. Speaker verification with multiple classifier fusion using Bayes based confidence measure. In *Proceedings of the 10th European Conference on Speech Communication and Technology (Interspeech 07)*, Antwerp, August 2007.

P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel. Joint factor analysis versus eigenchannels in speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4): 1435–1447, May 2007.

P. Kenny and P. Dumouchel. Experiments in speaker verification using factor analysis likelihood ratios. In *Proceedings of the Speaker and Language Recognition Workshop, Odyssey 2004*, Toledo, Spain, May 2004.

A. Kocsor, K. Kovács, and C. Szepesvári. Margin maximizing discriminant analysis. In *Proceedings of the 15th European Conference on Machine Learning*, Pisa, September 2004.

A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. MIT Press, 1995.

L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, 2004.

G. R. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

Y. Liu. *Negative Correlation Learning and Evolutionary Design of Neural Network Ensembles*. PhD thesis, University of New South Wales, New South Wales, Australia, 1999.

D. Matrouf, N. Scheffer, B. Fauve, and J.F. Bonastre. A straightforward and efficient implementation of the factor analysis model for speaker verification. In *Proceedings of the 10th European Conference on Speech Communication and Technology (Interspeech 07)*, Antwerp, August 2007.

N. C. Oza and K. Tumer. Input decimation ensembles: Decorrelation through dimensionality reduction. In *Lecture Notes in Computer Science*, volume 2096, pages 238–247. Springer, 2001.

P. Pavlidis, J. Cai, J. Weston, and W. S. Noble. Learning gene functional classifications from multiple data types. *Journal of Computational Biology*, 9:401–411, 2002.

D. Reynolds, W. Andrews, J. Campbell, J. Navratil, B. Peskin, A. Adami, Q. Jin, D. Klusacek, J. Abramson, R. Mihaescu, J. Godfrey, D. Jones, and B. Xiang. The SuperSID project: Exploiting high-level information for high-accuracy speaker recognition. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages 784–787, Hong Kong, April 2003.

D. Reynolds, W. Campbell, T. Gleason, C. Quillen, D. Sturim, and P. Torres-Carrasquillo. The 2004 MIT Lincoln Laboratory speaker recognition system. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Philadelphia, March 2005.

B. E. Rosen. Ensemble learning using decorrelated neural networks. *Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8:373–384, 1996.

K. Roy and P. Bhattacharya. Iris recognition with support vector machines. In *Advances in Biometrics*, volume 3832 of *Lecture Notes in Computer Science*, pages 486–492. Springer, 2005.

F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1): 1–47, 2002.

E. Shriberg, L. Ferrer, S. Kajarekar, A. Venkataraman, and A. Stolcke. Modeling prosodic feature sequences for speaker recognition. *Speech Communication*, 46(3-4):455–472, 2005. Special Issue on Quantitative Prosody Modelling for Natural Speech Description and Generation.

A. Smola and B. Schölkopf. A tutorial on support vector regression. *NeuroCOLT2 Technical Report NC2-TR-1998-030*, 1998.

A. Solomonoff, C. Quillen, and W. Campbell. Channel compensation for SVM speaker recognition. In *Proceedings of the Speaker and Language Recognition Workshop, Odyssey 2004*, Toledo, Spain, May 2004.

A. Stolcke, L. Ferrer, and S. Kajarekar. Improvements in MLLR-transform-based speaker recognition. In *Proceedings of the Speaker and Language Recognition Workshop, Odyssey 2006*, Puerto Rico, USA, June 2006.

A. Stolcke, S. S. Kajarekar, L. Ferrer, and E. Shriberg. Speaker recognition with session variability normalization based on MLLR adaptation transforms. *IEEE Transactions on Audio, Speech, and Language Processing*, September 2007.

K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29:341–348, 1996.

N. Ueda and R. Nakano. Generalization error of ensemble estimators. In *Proceedings of International Conference on Neural Networks*, pages 90–95, 1996.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1999.

M. Zanda, G. Brown, G. Fumera, and F. Roli. Ensemble learning in linearly combined classifiers via negative correlation. In *Proceedings of the International Workshop on Multiple Classifier Systems*, Prague, May 2007.

Journal of Machine Learning Research 10 (2009) 2115-2132 Submitted 12/06; Revised 2/08; Published 9/09

Deterministic Error Analysis of Support Vector Regression and Related Regularized Kernel Methods

Christian Rieger RIEGER@INS.UNI-BONN.DE
Institute for Numerical Simulation & Hausdorff Center for Mathematics
University of Bonn
Wegelerstr. 6, 53115 Bonn, Germany

Barbara Zwicknagl BARBARA.ZWICKNAGL@HCM.UNI-BONN.DE
Institute for Applied Mathematics & Hausdorff Center for Mathematics
University of Bonn
Endenicher Allee 60, 53115 Bonn, Germany

Editor: Bernhard Schölkopf

Abstract

We introduce a new technique for the analysis of kernel-based regression problems. The basic tools are sampling inequalities which apply to all machine learning problems involving penalty terms induced by kernels related to Sobolev spaces. They lead to explicit deterministic results concerning the worst case behaviour of ε- and ν-SVRs. Using these, we show how to adjust regularization parameters to get best possible approximation orders for regression. The results are illustrated by some numerical examples.

Keywords: sampling inequality, radial basis functions, approximation theory, reproducing kernel Hilbert space, Sobolev space

1. Introduction

Support Vector (SV) machines and related kernel-based algorithms are modern learning systems motivated by results of statistical learning theory as introduced by Vapnik (1995). The concept of SV machines is to provide a prediction function which is accurate on the given training data and which is sparse in the sense that it can be written in terms of a typically small subset of all samples, called the support vectors, as stated by Schölkopf et al. (1995). Therefore, SV regression and classification algorithms are closely related to regularized problems from classical approximation theory as pointed out by Girosi (1998) and Evgeniou et al. (2000) who had applied techniques from functional analysis to derive probabilistic error bounds for SV regression.
This paper provides a theoretical framework to derive deterministic error bounds for some popular SV machines. We show how a sampling inequality by Wendland and Rieger (2005) can be used to bound the worst-case generalization error for the ν- and the ε-regression without making any statistical assumptions on the inaccuracy of the training data. In contrast to the literature, our error bounds explicitly depend on the pointwise noise in the data. Thus they can be used for any subsequent probabilistic analysis modelling certain assumptions on the noise distribution.
The paper is organized as follows. In the next section we recall some basic facts about reproducing kernels in Hilbert spaces. Section 3 deals with regularized approximation problems in Hilbert spaces with reproducing kernels and outlines the connection to classical SV regression (SVR) al-

gorithms. We provide a deterministic error analysis for the ν- and the ε-SVR for both exact and inexact training data. Our analytical results showing optimal convergence orders in Sobolev spaces are illustrated by numerical experiments.

2. Reproducing Kernels in Hilbert Spaces

We suppose that K is a positive definite kernel on some domain $\Omega \subset \mathbb{R}^d$ which should contain at least one point. To start with, we briefly recall the well known definition of a reproducing kernel in a Hilbert space. In the following we shall use the notation that bold letters denote vectors, that is $\mathbf{v} = (v_1, \ldots, v_d)^T \in \mathbb{R}^d$.

Definition 1 *Let $\mathcal{H}(\Omega)$ be a Hilbert space of functions $f : \Omega \to \mathbb{R}$. A function $K : \Omega \times \Omega \to \mathbb{R}$ is called reproducing kernel of $\mathcal{H}(\Omega)$, if*

- *$K(\mathbf{y}, \cdot) \in \mathcal{H}(\Omega)$ for all $\mathbf{y} \in \Omega$ and*
- *$f(\mathbf{y}) = (f, K(\mathbf{y}, \cdot))_{\mathcal{H}(\Omega)}$ for all $f \in \mathcal{H}(\Omega)$ and all $\mathbf{y} \in \Omega$.*

For each positive definite kernel $K : \Omega \times \Omega \to \mathbb{R}$ there exists a unique Hilbert space $\mathcal{N}_K(\Omega)$ of functions $f : \Omega \to \mathbb{R}$, such that K is the reproducing kernel of $\mathcal{N}_K(\Omega)$ (see Wendland, 2005, Theorems 10.1 and 10.11). This Hilbert space $\mathcal{N}_K(\Omega)$ is called the *native space of K*. Though this definition of a native space is rather abstract, it can be shown that in some cases the native spaces coincide with classical function spaces.
From now on we shall only consider *radial* kernels K, that is,

$$K(\mathbf{x}, \mathbf{y}) = K(\|\mathbf{x} - \mathbf{y}\|) \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d ,$$

where we use the same notation for the kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and for the function $K : \mathbb{R}^d \to \mathbb{R}$. We hope that this does not cause any confusion. We shall mainly focus on continuous kernels $K \in L_1(\Omega)$, that is,

$$\|K\|_{L_1(\Omega)} := \int_\Omega |K(\mathbf{x})| \, d\mathbf{x} < \infty .$$

For $K \in L_1(\mathbb{R}^d)$, we define the Fourier transform $\hat{K}$ by

$$\hat{K}(\omega) := (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} K(\mathbf{x}) \, e^{-i\mathbf{x}\cdot\omega} d\mathbf{x} , \ \omega \in \mathbb{R}^d .$$

For the case $\Omega = \mathbb{R}^d$ there is the following characterization of native spaces of certain radial kernels $K : \Omega \to \mathbb{R}^d$ (Wendland, 2005, Theorem 10.12).

Theorem 2 *Suppose that $K \in C(\mathbb{R}^d) \cap L_1(\mathbb{R}^d)$ is a real-valued and positive definite radial kernel. Then the native space of K is given by*

$$\begin{aligned} \mathcal{N}_K(\mathbb{R}^d) &= \left\{ f \in L_2(\mathbb{R}^d) \cap C(\mathbb{R}^d) \ : \ \frac{\hat{f}}{\sqrt{\hat{K}}} \in L_2(\mathbb{R}^d) \right\} , \\ (f, g)_{\mathcal{N}_K(\mathbb{R}^d)} &= (2\pi)^{-d/2} \left(\frac{\hat{f}}{\sqrt{\hat{K}}}, \frac{\hat{g}}{\sqrt{\hat{K}}} \right)_{L_2(\mathbb{R}^d)} , \end{aligned}$$

where $\hat{f}$ denotes the Fourier transform of f.

We recall that the Sobolev spaces $W_2^s(\mathbb{R}^d)$ on $\mathbb{R}^d$ with $s \geq 0$ are given by

$$W_2^s(\mathbb{R}^d) := \left\{ f \in L_2(\mathbb{R}^d) \ : \ \hat{f}(\cdot)(1+\|\cdot\|_2^2)^{s/2} \in L_2(\mathbb{R}^d) \right\} . \tag{1}$$

Therefore for a radial kernel function K whose Fourier transform decays like

$$c_1(1+\|\cdot\|_2^2)^s \leq \hat{K} \leq c_2(1+\|\cdot\|_2^2)^s \ \ , s > d/2 \tag{2}$$

for some constants $c_1, c_2 > 0$, the associated native space $\mathcal{N}_K(\mathbb{R}^d)$ is $W_2^s(\mathbb{R}^d)$ with an equivalent norm. There are several examples of kernels satisfying the condition (2). One famous example for fixed $s \in (d/2, \infty)$ is the *Matern kernel* (Wendland, 2005)

$$K_s(\mathbf{x}) := \frac{2^{1-s}}{\Gamma(s)} \|\mathbf{x}\|_2^{s-d/2} \, \mathcal{K}_{d/2-s}(\|\mathbf{x}\|_2) ,$$

where $\mathcal{K}$ denotes the Bessel function of the third kind. In our examples, however, we focus on *Wendland's functions* (Wendland, 2005). They are very convenient to implement since they are compactly supported and piecewise polynomials. Such nice reproducing kernels are so far only available for certain choices of the space dimension d and the decay parameter s (see Wendland, 2005), but a recent result by Schaback (2009) covers almost all cases of practical interest. We shall explain some more properties of these kernels in the experimental part, see Section 10, and refer to the recent monograph by Wendland (2005) for details.
In order to establish the equivalence of native spaces and Sobolev spaces on bounded domains one needs certain extension theorems for Sobolev functions on bounded domains (see Wendland, 2005).

Definition 3 *Let $\Omega \subset \mathbb{R}^d$ be a domain. We define the Sobolev spaces of integer orders $k \in \mathbb{N}$ as*

$$W_2^k(\Omega) = \{ f \in L_2(\Omega) \ : \ f \text{ has weak derivatives } D^\alpha f \in L_2(\Omega) \text{ of order } |\alpha| \leq k \}$$

with the norm

$$\|u\|_{W_2^k(\Omega)} := \left(\sum_{|\alpha| \leq k} \|D^\alpha u\|_{L_2(\Omega)}^2 \right)^{1/2} .$$

For fractional smoothness $s = k + \sigma$ with $0 < \sigma < 1$ and $k \in \mathbb{N}$ we define the semi-norm

$$|u|_{W_2^s(\Omega)} := \left(\sum_{|\alpha|=k} \int_\Omega \int_\Omega \frac{|D^\alpha u(\mathbf{x}) - D^\alpha u(\mathbf{y})|^2}{\|\mathbf{x}-\mathbf{y}\|_2^{d+2\sigma}} d\mathbf{x} d\mathbf{y} \right)^{1/2} ,$$

and set

$$W_2^s(\Omega) := \left\{ u \in L_2(\Omega) \ : \ \left(\|u\|_{W_2^k(\Omega)}^2 + |u|_{W_2^s(\Omega)}^2 \right)^{1/2} < \infty \right\} .$$

In the case $\Omega = \mathbb{R}^d$ this space is known to be equivalent to the space given by (1) in terms of Fourier transforms (for more details on these spaces, see Wloka, 1982). Finally, Wendland (2005) proves the following equivalence for domains having Lipschitz boundaries. Roughly speaking, a set $\Omega \subset \mathbb{R}^d$ has a Lipschitz boundary if its boundary is locally (in a suitable direction) the graph of a Lipschitz function such that Ω lies completely on one hand-side of this graph (see Brenner and Scott, 1994). Then there is the following theorem (see Wendland, 2005, Cor. 10.48).

Theorem 4 *Suppose that $K \in L_1(\mathbb{R}^d)$ has a Fourier transform that decays as $(1+\|\cdot\|_2^2)^{-s}$ for $s > d/2$. Suppose that Ω has a Lipschitz boundary. Then*

$$\mathcal{N}_K(\Omega) \cong W_2^s(\Omega)$$

with equivalent norms.

3. Regularized Problems in Native Hilbert Spaces

In the native Hilbert spaces we consider the following learning or recovery problem. We assume that we are given (possibly only approximate) function values $y_1,\dots,y_N \in \mathbb{R}$ of an unknown function $f \in \mathcal{N}_K(\Omega)$ on some scattered points $X := \{\mathbf{x}^{(1)},\dots,\mathbf{x}^{(N)}\} \subset \Omega$, that is $f(\mathbf{x}^{(j)}) \approx y_j$ for $j = 1,\dots,N$. In the following we shall use the notation that bold letters denote vectors, that is $\mathbf{v} = (v_1,\dots,v_d)^T \in \mathbb{R}^d$.

To control accuracy and complexity of the reconstruction simultaneously, we use the optimization problem

$$\min_{\substack{s\in\mathcal{N}_K(\Omega)\\ \varepsilon\in\mathbb{R}^+}} \frac{1}{N}\sum_{j=1}^{N} V_\varepsilon\left(\left|s\left(\mathbf{x}^{(j)}\right) - y_j\right|\right) + \frac{1}{2C}\|s\|^2_{\mathcal{N}_K(\Omega)}\ , \tag{3}$$

where $C > 0$ is a positive parameter and V_ε denotes a positive function which may be parametrized by a positive real number ε. We point out that V_ε need not be a classical loss function. Therefore we shall give some proofs of results which were formulated by Schölkopf and Smola (2002) in the case of V_ε being a loss function.

Theorem 5 (Representer theorem) *If $(s_{X,\mathbf{y}}, \varepsilon^*)$ is a solution of the optimization problem (3), then there exists a vector $\mathbf{w} \in \mathbb{R}^N$ such that*

$$s_{X,\mathbf{y}}(\cdot) = \sum_{j=1}^{N} w_j K\left(\mathbf{x}^{(j)},\cdot\right)\ ,$$

that is $s_{X,\mathbf{y}} \in span\left\{K\left(\mathbf{x}^{(1)},\cdot\right),\dots,K\left(\mathbf{x}^{(N)},\cdot\right)\right\}$.

Proof For the readers' convenience, we repeat the proof from Schölkopf and Smola (2002) in our specific situation. Every $s \in \mathcal{N}_K(\Omega)$ can be decomposed into two parts $s = s_{||} + s_\perp$, where $s_{||}$ is contained in the linear span of $\left\{K\left(\mathbf{x}^{(1)},\cdot\right),\dots,K\left(\mathbf{x}^{(N)},\cdot\right)\right\}$, and $s_\perp$ is contained in the orthogonal complement, that is $\left\langle s_{||}, s_\perp\right\rangle_{\mathcal{N}_K(\Omega)} = 0$. By the reproducing property of the kernel K in the native space, the problem (3) can be rewritten as

$$\min_{\substack{s=s_{||}+s_\perp\\ \varepsilon\in\mathbb{R}^+}} \frac{1}{N}\sum_{j=1}^{N} V_\varepsilon\left(\left|\left\langle s_{||}, K\left(\mathbf{x}^{(j)},\cdot\right)\right\rangle - y_j\right|\right) + \frac{1}{2C}\left\|s_{||}\right\|^2_{\mathcal{N}_K(\Omega)} + \frac{1}{2C}\|s_\perp\|^2_{\mathcal{N}_K(\Omega)}\ .$$

Therefore a solution $(s_{X,\mathbf{y}}, \varepsilon^*)$ of the optimization problem (3) satisfies $(s_{X,\mathbf{y}})_\perp = 0$, which implies $s_{X,\mathbf{y}} \in \operatorname{span}\left\{K\left(\mathbf{x}^{(1)},\cdot\right),\dots,K\left(\mathbf{x}^{(N)},\cdot\right)\right\}$. ■

Since the proof of Theorem 5 does not depend on the minimality with respect to ε this result holds also true if ε is a fixed parameter instead of a primal variable. To be precise we state this result as a corollary.

Corollary 6 *If $s_{X,\mathbf{y}}$ is a solution of the optimization problem*

$$\min_{s\in\mathcal{N}_K(\Omega)} \frac{1}{N}\sum_{j=1}^{N} V_\varepsilon\left(\left|s\left(\mathbf{x}^{(j)}\right)-y_j\right|\right)+\frac{1}{2C}\|s\|^2_{\mathcal{N}_K(\Omega)}\ , \tag{4}$$

with $\varepsilon\in\mathbb{R}^+$ being a fixed parameter, then $s_{X,\mathbf{y}}\in span\left\{K\left(\mathbf{x}^{(1)},\cdot\right),\ldots,K\left(\mathbf{x}^{(N)},\cdot\right)\right\}$.

The representer theorems can be used to reformulate infinite-dimensional optimization problems of the forms (3) or (4) in a finite-dimensional setting (see Schölkopf and Smola, 2002).

4. Support Vector Regression

As a first optimization problem of the form (3) we consider the ν-SVR which was introduced by Schölkopf et al. (2000). The function $V_\varepsilon(\mathbf{x})=|\mathbf{x}|_\varepsilon+\varepsilon\nu$ is related to Vapnik's ε-intensive loss function (Vapnik, 1995)

$$|\mathbf{x}|_\varepsilon=\begin{cases}0 & if\ |\mathbf{x}|\le\varepsilon\\ |\mathbf{x}|-\varepsilon & if\ |\mathbf{x}|>\varepsilon\end{cases}\ ,$$

but has an additional term with a positive parameter ν. The associated optimization problem is called ν-SVR and takes the form

$$\min_{\substack{s\in\mathcal{N}_K(\Omega)\\ \varepsilon\in\mathbb{R}^+}} \frac{1}{N}\sum_{j=1}^{N}\left|s\left(\mathbf{x}^{(j)}\right)-y_j\right|_\varepsilon+\varepsilon\nu+\frac{1}{2C}\|s\|^2_{\mathcal{N}_K(\Omega)}\ . \tag{5}$$

Theorem 7 *The optimization problem (5) possesses a solution $\left(s^{(\nu)}_{X,\mathbf{y}},\varepsilon^*\right)$.*

Proof This follows from a general result by Micchelli and Pontil (2005). The problem (5) is equivalent to the optimization problem

$$\min_{\substack{s\in\mathcal{N}_K(\Omega)\\ \delta\in\mathbb{R}}} \frac{1}{N}\sum_{j=1}^{N}\left|s\left(\mathbf{x}^{(j)}\right)-y_j\right|_{\delta^2}+\delta^2\nu+\frac{1}{2C}\|s\|^2_{\mathcal{N}_K(\Omega)}\ . \tag{6}$$

If we set $\mathcal{H}:=\mathcal{N}_K(\Omega)\times\mathbb{R}$ we can define an inner product on $\mathcal{H}$ by

$$\langle h_1,h_2\rangle_{\mathcal{H}}:=\langle f_1,f_2\rangle_{\mathcal{N}_K(\Omega)}+2C\nu\langle r_1,r_2\rangle_{\mathbb{R}}$$

for $h_j=(f_j,r_j)$, $j=1,2$. To make $\mathcal{H}$ a space of functions we use the canonical identification of $\mathbb{R}$ with the space of constant functions $\mathbb{R}\to\mathbb{R}$. The Hilbert space $\mathcal{H}$ then has the reproducing kernel $\tilde{K}:=\left(K,\frac{1}{2C\nu}\mathbf{1}\right)$ where $\mathbf{1}$ denotes the constant function which maps everything to 1, that is $\tilde{K}((\mathbf{x},r),(\mathbf{y},s))=K(\mathbf{x},\mathbf{y})+1/(2C\nu)$ for all $r,s\in\mathbb{R}$. With this notation the problem (6) can be rewritten as

$$\min_{(s,\delta)\in\mathcal{H}} Q^{\mathbf{y}}\left(I_X(s,\delta)\right)+\frac{1}{2C}\|(s,\delta)\|^2_{\mathcal{H}}\ , \tag{7}$$

where

$$I_X(s,\delta):=\left(s(\mathbf{x}^{(1)}),\ldots,s(\mathbf{x}^{(N)}),\delta\right)^T\in\mathbb{R}^{N+1}$$

and

$$Q^{\mathbf{y}} : \mathbb{R}^{N+1} \to \mathbb{R}, \quad Q^{\mathbf{y}}(\mathbf{p}, \delta) = \frac{1}{N} \sum_{j=1}^{N} \left| p_j - y_j \right|_{\delta^2} .$$

Since $Q^{\mathbf{y}}$ is continuous on $\mathbb{R}^{N+1}$ for all $\mathbf{y} \in \mathbb{R}^N$, the problem (7) possesses a solution as shown by Micchelli and Pontil (2005). ■

If we introduce the slack variables $\xi, \xi^* \in \mathbb{R}^N$, the representer theorem gives us an equivalent finite-dimensional problem which was considered by Schölkopf et al. (2000).

$$\min_{\substack{\mathbf{w} \in \mathbb{R}^{\mathbf{N}} \\ \xi^*, \xi \in \mathbb{R}^N \\ \varepsilon \in \mathbb{R}^+}} \frac{1}{2} \mathbf{w}^T \mathbf{K} \mathbf{w} + C \left(\nu\varepsilon + \frac{1}{N} \sum_{j=1}^{N} \left(\xi_j + \xi_j^* \right) \right)$$

$$\begin{aligned} \text{subject to} \quad (\mathbf{K}\mathbf{w})_j - y_j &\leq \varepsilon + \xi_j , \\ (-\mathbf{K}\mathbf{w})_j + y_j &\leq \varepsilon + \xi_j^* , \\ \xi_j^*, \xi_j \geq 0 , \qquad \varepsilon &\geq 0 \quad \text{for } 1 \leq j \leq N , \end{aligned} \tag{8}$$

where

$$\mathbf{K} = \left(K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \right) \right)_{i,j=1\ldots N}$$

denotes the Gram matrix of the kernel K. We will use this equivalent problem for implementation and our numerical tests.

A particularly interesting problem arises if we skip the parameter ν and let ε be fixed. Then the optimization problem (8) takes the form

$$\min_{\substack{\mathbf{w} \in \mathbb{R}^{\mathbf{N}} \\ \xi^*, \xi \in \mathbb{R}^N}} \frac{1}{2} \mathbf{w}^T \mathbf{K} \mathbf{w} + C \frac{1}{N} \sum_{j=1}^{N} \left(\xi_j + \xi_j^* \right)$$

$$\begin{aligned} \text{subject to} \quad (\mathbf{K}\mathbf{w})_j - y_j &\leq \varepsilon + \xi_j , \\ (-\mathbf{K}\mathbf{w})_j + y_j &\leq \varepsilon + \xi_j^* , \\ \xi_j^*, \xi_j &\geq 0 \quad \text{for } 1 \leq j \leq N . \end{aligned} \tag{9}$$

Schölkopf et al. (2000) called this problem ε-SVR. Similarly to the ν-SVR, the problem (9) can be formulated as a regularized minimization problem in a Hilbert space (Evgeniou et al., 2000), namely

$$\min_{s \in \mathcal{N}_K(\Omega)} \frac{1}{N} \sum_{j=1}^{N} \left| s\left(\mathbf{x}^{(j)} \right) - y_j \right|_{\varepsilon} + \frac{1}{2C} \left\| s \right\|_{\mathcal{N}_K(\Omega)}^2 . \tag{10}$$

Like the ν-SVR, this optimization problem possesses a solution (see Micchelli and Pontil, 2005, Lemma 1).

5. A Sampling Inequality

We shall employ a special case of a *sampling inequality* introduced by Wendland and Rieger (2005). It requires the following assumptions which we need from now on. Let $\Omega \subset \mathbb{R}^d$ be a bounded

domain with Lipschitz boundary that satisfies an interior cone condition. A domain Ω is said to satisfy an interior cone condition with radius $r > 0$ and angle $\theta \in \left(0, \frac{\pi}{2}\right)$ if for every $\mathbf{x} \in \Omega$ there is a unit vector $\xi(\mathbf{x})$ such that the cone

$$C(\mathbf{x}, \xi(\mathbf{x}), \theta, r) := \left\{ \mathbf{x} + \lambda \mathbf{y} \ : \ \mathbf{y} \in \mathbb{R}^d, \|\mathbf{y}\|_2 = 1, \mathbf{y}^T \xi(\mathbf{x}) \geq \cos(\theta), \lambda \in [0, r] \right\}$$

is contained in Ω. In particular, a domain which satisfies an interior cone condition cannot have any outward cusps. We shall assume for the rest of this paper that Ω satisfies an interior cone condition with radius $R_{\max}$ and angle θ. We shall derive estimates that are valid only if the training points are sufficiently dense in Ω. To make this condition precise, we will need a slightly unhandy constant which depends only on the geometry of Ω, namely (see Wendland, 2005)

$$C_\Omega := \frac{\sin\left(2\arcsin\left(\frac{\sin\theta}{4(1+\sin\theta)}\right)\right)\sin\theta}{8\left(1+\sin\left(2\arcsin\left(\frac{\sin\theta}{4(1+\sin\theta)}\right)\right)\right)(1+\sin\theta)} R_{\max} \ .$$

Suppose that K is a radial kernel function such that the native Hilbert space of K is norm-equivalent to a Sobolev space, that is $\mathcal{N}_K(\Omega) = W_2^\tau(\Omega)$. Here we assume that $\lfloor \tau - \frac{1}{2} \rfloor > d/2$, where we use the notation $\lfloor t \rfloor := \max\{n \in \mathbb{N}_0 : n \leq t\}$ for $t \geq 0$. Furthermore, let $X = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\} \subset \Omega$ be a finite set with sufficiently small fill distance

$$h := h_{X,\Omega} := \sup_{\mathbf{x} \in \Omega} \min_{\mathbf{x}^{(j)} \in X} \left\| \mathbf{x} - \mathbf{x}^{(j)} \right\|_2 \ .$$

The fill distance can be interpreted geometrically as the radius of the largest ball with center in $\bar{\Omega}$ that does not contain any of the points $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$. It is a useful quantity for the deterministic error analysis in Sobolev spaces. The case $h = 0$ implies that $X = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\}$ is dense in Ω, and therefore convergence is studied for the limit $h \to 0$ which means that the domain Ω is equally filled with points from X. Let us explain the relation to the usual error bounds in terms of the number of points N. In the case of regularly distributed points we have that $h = cN^{-\frac{1}{d}}$ with some constant $c > 0$ (Wendland, 2005). Therefore the limit $h \to 0$ is equivalent to the limit $N \to \infty$ which is the more intuitive meaning of asymptotic convergence. But there is a drawback, since the error bounds in terms of N depend crucially on the space dimension d, while error bounds in terms of the fill distance h are dominated by the smoothness of the function to be learned. We will comment on this again later for the special error bounds we consider here. We shall use the following result by Wendland and Rieger (2005).

Theorem 8 *Suppose $\Omega \subset \mathbb{R}^d$ is a bounded domain with Lipschitz boundary that satisfies an interior cone condition. Let τ be a positive real number with $\lfloor \tau - \frac{1}{2} \rfloor > \frac{d}{2}$, and let $1 \leq q \leq \infty$. Then there exists a positive constant $C > 0$ such that for all discrete sets $X \subset \Omega$ with sufficiently small fill distance $h := h_{X,\Omega} \leq C_\Omega \tau^{-2}$ the inequality*

$$\|u\|_{L_q(\Omega)} \leq C\left(h^{\tau - d\left(\frac{1}{2} - \frac{1}{q}\right)_+} \|u\|_{W_2^\tau(\Omega)} + \|u|_X\|_{\ell_\infty(X)}\right)$$

holds for all $u \in W_2^\tau(\Omega)$, where we use the notation $(t)_+ := \max\{0, t\}$.

We shall apply this theorem to the residual function $f - s_{X,\mathbf{y}}$ of the function $f \in W_2^\tau(\Omega)$ to be recovered and a solution $s_{X,\mathbf{y}} \in W_2^\tau(\Omega)$ of the regression problem. In our applications we shall focus on the two main cases $q = \infty$ and $q = 2$. Other cases can be treated analogously. It will turn out that we get optimal convergence rates in the noiseless case. In presence of noise the resulting error will explicitly be bounded in terms of the noise in the data.

6. ν-SVR with Exact Data

In order to derive error bounds for the ν-SVR optimization problem (5) we shall apply Theorem 8 to the residual $f - s^{(\nu)}_{X,\mathbf{y}}$, where $\left(s^{(\nu)}_{X,\mathbf{y}}, \varepsilon^*\right)$ denotes a solution to the problem (5) for $X := \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\} \subset \Omega$ and $\mathbf{y} \in \mathbb{R}^N$. In this section we consider exact data, that is

$$f\left(\mathbf{x}^{(j)}\right) = y_j \quad \text{for } j = 1, \ldots, N \tag{11}$$

for a function $f \in W_2^\tau(\Omega) \cong \mathcal{N}_K(\Omega)$. As pointed out by Wendland and Rieger (2005) we first need a stability and a consistency estimate for the solution $s^{(\nu)}_{X,\mathbf{y}}$.

Lemma 9 *Under the assumption (11) concerning the data, we find that for every X a solution $\left(s^{(\nu)}_{X,\mathbf{y}}, \varepsilon^*\right)$ to problem (5) satisfies*

$$\begin{aligned} \left\|s^{(\nu)}_{X,\mathbf{y}}\right\|_{\mathcal{N}_K(\Omega)} &\leq \|f\|_{\mathcal{N}_K(\Omega)} \quad \textit{and} \\ \left\|s^{(\nu)}_{X,\mathbf{y}}|_X - \mathbf{y}\right\|_{\ell_\infty(X)} &\leq \frac{N}{2C}\|f\|^2_{\mathcal{N}_K(\Omega)} + \varepsilon^* \cdot (1 - N\nu) \ . \end{aligned}$$

Proof We denote the objective function of the optimization problem (5) by

$$H^{\mathbf{y}}_{C,\nu}(s,\varepsilon) := \frac{1}{N}\sum_{j=1}^{N}\left|s\left(\mathbf{x}^{(j)}\right) - y_j\right|_\varepsilon + \nu\varepsilon + \frac{1}{2C}\|s\|^2_{\mathcal{N}_K(\Omega)} \ , \tag{12}$$

and the interpolant to f with respect to X and K with I_f, that is $I_f|_X = \mathbf{y}$ and $I_f \in \operatorname{span}\left\{K\left(\mathbf{x}^{(1)}, \cdot\right), \ldots, K\left(\mathbf{x}^{(N)}, \cdot\right)\right\}$. With this notation we have

$$\frac{1}{2C}\left\|s^{(\nu)}_{X,\mathbf{y}}\right\|^2_{\mathcal{N}_K(\Omega)} \leq H^{\mathbf{y}}_{C,\nu}\left(s^{(\nu)}_{X,\mathbf{y}}, \varepsilon^*\right) \leq H^{\mathbf{y}}_{C,\nu}(I_f, 0) = \frac{1}{2C}\left\|I_f\right\|^2_{\mathcal{N}_K(\Omega)} \leq \frac{1}{2C}\|f\|^2_{\mathcal{N}_K(\Omega)}$$

since $\left\|I_f\right\|_{\mathcal{N}_K(\Omega)} \leq \|f\|_{\mathcal{N}_K(\Omega)}$ (Wendland, 2005), which implies the first claim.
Furthermore we have for $i = 1, \ldots, N$

$$\begin{aligned} \left|s^{(\nu)}_{X,\mathbf{y}}\left(\mathbf{x}^{(i)}\right) - y_i\right| &\leq \sum_{j=1}^{N}\left|s^{(\nu)}_{X,\mathbf{y}}\left(\mathbf{x}^{(j)}\right) - y_j\right|_{\varepsilon^*} + \varepsilon^* \leq N H^{\mathbf{y}}_{C,\nu}\left(s^{(\nu)}_{X,\mathbf{y}}, \varepsilon^*\right) + \varepsilon^*(1 - N\nu) \\ &\leq N H^{\mathbf{y}}_{C,\nu}(I_f, 0) + \varepsilon^*(1 - N\nu) \leq \frac{N}{2C}\left\|I_f\right\|^2_{\mathcal{N}_K(\Omega)} + \varepsilon^*(1 - N\nu) \\ &\leq \frac{N}{2C}\|f\|^2_{\mathcal{N}_K(\Omega)} + \varepsilon^*(1 - N\nu) \ , \end{aligned}$$

which finishes the proof. ■

With Theorem 8 we find immediately the following result.

Theorem 10 *Suppose $\Omega \subset \mathbb{R}^d$ is a bounded domain with Lipschitz boundary that satisfies an interior cone condition. Let τ be a positive real number with $\lfloor \tau - \frac{1}{2} \rfloor > \frac{d}{2}$ and $1 \leq q \leq \infty$. We suppose*

$f \in W_2^\tau(\Omega)$ with $f(\mathbf{x}^{(i)}) = y_i$. Let $\left(s_{X,\mathbf{y}}^{(\nu)}, \varepsilon^*\right)$ be a solution of the ν-SVR. Then there is a constant $\tilde{C} > 0$, which depends on τ, d and Ω but not on f or X, such that the approximation error can be bounded by

$$\left\| f - s_{X,\mathbf{y}}^{(\nu)} \right\|_{L_q(\Omega)} \leq \tilde{C} \left(2h^{\tau - d\left(\frac{1}{2} - \frac{1}{q}\right)_+} \|f\|_{W_2^\tau(\Omega)} + \frac{N}{2C} \|f\|^2_{W_2^\tau(\Omega)} + (1 - N\nu) \cdot \varepsilon^* \right)$$

for all discrete sets $X \subset \Omega$ with fill distance $h := h_{X,\Omega} \leq C_\Omega \tau^{-2}$.

Proof Combining Lemma 9 and Theorem 8 leads to

$$\begin{aligned}
\left\| f - s_{X,\mathbf{y}}^{(\nu)} \right\|_{L_q(\Omega)} &\leq \tilde{C} \left(h^{\tau - d\left(\frac{1}{2} - \frac{1}{q}\right)_+} \left\| f - s_{X,\mathbf{y}}^{(\nu)} \right\|_{W_2^\tau(\Omega)} + \left\| \mathbf{y} - s_{X,\mathbf{y}}^{(\nu)}|_X \right\|_{\ell_\infty(X)} \right) \\
&\leq \tilde{C} \left(h^{\tau - \left(\frac{d}{2} - \frac{d}{q}\right)_+} \left(\|f\|_{W_2^\tau(\Omega)} + \left\| s_{X,\mathbf{y}}^{(\nu)} \right\|_{W_2^\tau(\Omega)} \right) + \left\| \mathbf{y} - s_{X,\mathbf{y}}^{(\nu)}|_X \right\|_{\ell_\infty(X)} \right) \\
&\leq \tilde{C} \left(2h^{\tau - \left(\frac{d}{2} - \frac{d}{q}\right)_+} \|f\|_{W_2^\tau(\Omega)} + \frac{N}{2C} \|f\|^2_{W_2^\tau(\Omega)} + (1 - N\nu)\, \varepsilon^* \right) .
\end{aligned}$$

■

At first glance the term containing ε^* seems to be odd because it could be uncontrollable. But according to Chang and Lin (2002) we can at least assume ε^* to be bounded by

$$\varepsilon^* \leq \frac{1}{2} \left(\max_{i=1,\ldots,N} y_i - \min_{i=1,\ldots,N} y_i \right) .$$

If this inequality is not satisfied, the problem (8) possesses only the trivial solution $s \equiv 0$ which is not interesting. Furthermore, we see that the ε^*-term occurs with a factor $(1 - N\nu)$, which can be used to control this term. If we choose $\nu \geq \frac{1}{N}$, the term $(1 - N\nu)\,\varepsilon^*$ vanishes or is even negative. The parameter ν is a lower bound on the fraction of support vectors (see Schölkopf et al., 2000), and hence $\nu = 1/N$ means to get at least one support vector, that is a non-trivial solution. Since we are not interested in the case of trivial solutions, the condition $\nu \geq 1/N$ is a reasonable assumption. On the other hand, we can use the results from Lemma 9 to derive a more explicit upper bound on $\varepsilon^* = \varepsilon^*(C, \nu, f)$ by

$$0 \leq \left\| s_{X,\mathbf{y}}^{(\nu)}|_X - \mathbf{y} \right\|_{\ell_\infty(X)} \leq \frac{N}{2C} \|f\|^2_{\mathcal{N}_K(\Omega)} + \varepsilon^* (1 - N\nu) .$$

If we assume $\nu > 1/N$, this leads to

$$\varepsilon^* = \varepsilon^*(C, \nu, f) \leq \frac{N}{2C(N\nu - 1)} \|f\|^2_{\mathcal{N}_K(\Omega)} .$$

Note that these bounds cannot be used for a better parameter choice, since we would need to rearrange this inequality and solve for C or ν. This would only be possible if there were lower bounds on ε^* as well. Moreover, the parameter C appears in our error bound as a factor $\frac{N}{2C}$ which implies that we expect convergence only in the case $C \to \infty$. In this case ε^* will be small, as can be deduced from problem (8).

We shall now make our bounds more explicit for the case of quasi-uniformly distributed points. In this case the number of points N and the fill distance h are related to each other by

$$c_1 N^{-1/d} \leq h \leq c_2 N^{-1/d} \,, \tag{13}$$

where c_1 and c_2 denote positive constants (see Wendland, 2005, Proposition 14.1).

Corollary 11 *In case of quasi-uniform exact data we can choose the problem parameters as*

$$C = \frac{N \|f\|_{W_2^\tau(\Omega)}}{2h^\tau} \approx h^{-(\tau+d)} \|f\|_{W_2^\tau(\Omega)} \text{ and } \nu \geq \frac{1}{N}$$

to get

$$\left\| f - s^{(\nu)}_{X,\mathbf{y}} \right\|_{L_2(\Omega)} \leq \tilde{C} h^\tau \|f\|_{W_2^\tau(\Omega)} \leq \tilde{C} N^{-\frac{\tau}{d}} \|f\|_{W_2^\tau(\Omega)} \,;$$

or as

$$C = \frac{N \|f\|_{W_2^\tau(\Omega)}}{2h^{\tau-\frac{d}{2}}} \approx h^{-\left(\tau+\frac{d}{2}\right)} \|f\|_{W_2^\tau(\Omega)} \text{ and } \nu \geq \frac{1}{N}$$

to get

$$\left\| f - s^{(\nu)}_{X,\mathbf{y}} \right\|_{L_\infty(\Omega)} \leq \tilde{C} h^{\tau-\frac{d}{2}} \|f\|_{W_2^\tau(\Omega)} \leq \tilde{C} N^{-\frac{\tau}{d}+\frac{1}{2}} \|f\|_{W_2^\tau(\Omega)}$$

for all discrete sets $X \subset \Omega$ with fill distance $h := h_{X,\Omega} \leq C_\Omega \tau^{-2}$, with generic positive constants $\tilde{C}$ which depend on τ, d, Ω but not on f or X.

Note that these bounds yield arbitrarily high convergence orders, provided that the functions are smooth enough, that is τ is large enough. Therefore they are in this setting better than the usual minimax rate $N^{-\frac{2\tau}{2\tau+d}}$ (see Stone, 1982). In the following we shall only give our error estimates in terms of the fill distance h rather than in terms of the number of points N. This is due to the fact that the approximation rate τ in h is independent of the space dimension d. However it should be clear how the approximation rates translate into error estimates in terms of N in the case of quasi-uniform data due to the inequality (13). Note that the parameter choice in the case of arbitrary, non-uniformly distributed data can be treated analogously.
Corollary 11 shows, that the solution of the ν-SVR leads to the same approximation orders with respect to the fill distance h as classical kernel-based interpolation (see Wendland, 2005). But the ν-SVR allows for much more flexibility and less complicated solutions. Our numerical results will confirm these convergence rates.

7. ν-SVR with Inexact Data

In this section we denote again by $\left(s^{(\nu)}_{X,\mathbf{y}}, \varepsilon^*\right)$ the solution to the problem (5) for a set of points $X := \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\} \subset \Omega$ and $\mathbf{y} \in \mathbb{R}^N$, but we allow the given data to be corrupted by some additive error $\mathbf{r} = (r_1, \ldots, r_N)$, that means

$$f\left(\mathbf{x}^{(j)}\right) = y_j + r_j \quad \text{for } j = 1, \ldots, N, \tag{14}$$

where is $f \in W_2^\tau(\Omega) \cong \mathcal{N}_K(\Omega)$. Note that there are no assumptions concerning the error distribution. As in the previous section we have to show a stability and a consistency estimate of the following form.

Lemma 12 *Under the assumption (14) concerning the data* $\mathbf{y}$*, a solution* $\left(s^{(\nu)}_{X,\mathbf{y}}, \varepsilon^*\right)$ *to the optimization problem (5) satisfies for every* X *and for all* $\varepsilon \geq 0$

$$\begin{aligned}
\left\| s^{(\nu)}_{X,\mathbf{y}} \right\|_{\mathcal{N}_K(\Omega)} &\leq \sqrt{\frac{2C}{N} \sum_{j=1}^{N} \left| r_j \right|_\varepsilon + 2C\nu\varepsilon + \|f\|^2_{\mathcal{N}_K(\Omega)}} \quad \textit{and} \\
\left\| s^{(\nu)}_{X,\mathbf{y}} - \mathbf{y} \right\|_{\ell_\infty(X)} &\leq \sum_{j=1}^{N} \left| r_j \right|_\varepsilon + \nu N\varepsilon + (1 - N\nu)\,\varepsilon^* + \frac{N}{2C} \|f\|^2_{\mathcal{N}_K(\Omega)} .
\end{aligned}$$

Proof Again, we denote the interpolant to f with respect to X and K by I_f and use $H^{\mathbf{y}}_{C,\nu}$ as defined in Equation (12). Then we have for all $\varepsilon > 0$

$$\frac{1}{2C} \left\| s^{(\nu)}_{X,\mathbf{y}} \right\|^2_{\mathcal{N}_K(\Omega)} \leq H^{\mathbf{y}}_{C,\nu}\left(s^{(\nu)}_{X,\mathbf{y}}, \varepsilon^*\right) \leq H^{\mathbf{y}}_{C,\nu}(I_f, \varepsilon) \leq \frac{1}{N} \sum_{j=1}^{N} \left| r_j \right|_\varepsilon + \nu\varepsilon + \frac{1}{2C} \|f\|^2_{\mathcal{N}_K(\Omega)}$$

which implies

$$\left\| s^{(\nu)}_{X,\mathbf{y}} \right\|_{\mathcal{N}_K(\Omega)} \leq \sqrt{\frac{2C}{N} \sum_{j=1}^{N} \left| r_j \right|_\varepsilon + 2C\nu\varepsilon + \|f\|^2_{\mathcal{N}_K(\Omega)}} .$$

Moreover we have for all $i = 1, \ldots, N$ and all $\varepsilon > 0$

$$\begin{aligned}
\left| s^{(\nu)}_{X,\mathbf{y}}\left(\mathbf{x}^{(i)}\right) - y_i \right| &\leq \sum_{j=1}^{N} \left| s^{(\nu)}_{X,\mathbf{y}}\left(\mathbf{x}^{(j)}\right) - y_j \right|_{\varepsilon^*} + \varepsilon^* \\
&\leq N H^{\mathbf{y}}_{C,\nu}\left(s^{(\nu)}_{X,\mathbf{y}}, \varepsilon^*\right) + (1 - N\nu)\,\varepsilon^* \\
&\leq \sum_{j=1}^{N} \left| r_j \right|_\varepsilon + \nu N\varepsilon + (1 - N\nu)\,\varepsilon^* + \frac{N}{2C} \|f\|^2_{\mathcal{N}_K(\Omega)} .
\end{aligned}$$

■

Again we can use the results from Lemma 12 to derive a more explicit upper bound on $\varepsilon^* = \varepsilon^*(C, \nu, f, \varepsilon)$. Note that ε^* depends now also on the free parameter ε.

$$0 \leq \left\| s^{(\nu)}_{X,\mathbf{y}|X} - \mathbf{y} \right\|_{\ell_\infty(X)} \leq \frac{N}{2C} \|f\|^2_{\mathcal{N}_K(\Omega)} + \varepsilon^*(1 - N\nu) + \sum_{j=1}^{N} \left| r_j \right|_\varepsilon + \nu N\varepsilon .$$

If we assume $\nu > 1/N$, this leads to

$$\varepsilon^*(C, \nu, f, \varepsilon) \leq \frac{1}{N\nu - 1} \left(\frac{N}{2C} \|f\|^2_{\mathcal{N}_K(\Omega)} + \sum_{j=1}^{N} \left| r_j \right|_\varepsilon + \nu N\varepsilon \right) .$$

Using the sampling inequality as in the case of exact data leads to the following result on L_q-norms.

Theorem 13 *We suppose $f \in W_2^\tau(\Omega)$ with $f\left(\mathbf{x}^{(i)}\right) = y_i + r_i$. Let $\left(s_{X,\mathbf{y}}^{(\nu)}, \varepsilon^*\right)$ be a solution of the ν-SVR, that is the optimization problem (5). Then there is a constant $\tilde{C} > 0$, which depends on τ, d and Ω but not on f or X, such that for all $\varepsilon > 0$ the approximation error can be bounded by*

$$\left\|f - s_{X,\mathbf{y}}^{(\nu)}\right\|_{L_q(\Omega)} \leq \tilde{C}\left(h^{\tau-\left(\frac{d}{2}-\frac{d}{q}\right)_+}\left(\|f\|_{W_2^\tau(\Omega)} + \sqrt{\frac{2C}{N}\sum_{j=1}^{N}\left|r_j\right|_\varepsilon + 2C\nu\varepsilon + \|f\|_{W_2^\tau(\Omega)}^2}\right)\right.$$
$$\left. + \sum_{j=1}^{N}\left|r_j\right|_\varepsilon + \nu N\varepsilon + \varepsilon^*(1-N\nu) + \frac{N}{2C}\|f\|_{W_2^\tau(\Omega)}^2 + \|\mathbf{r}\|_{\ell_\infty(X)}\right)$$

for all discrete sets $X \subset \Omega$ with fill distance $h := h_{X,\Omega} \leq C_\Omega \tau^{-2}$.

Note that the choice of the "optimal" ε leading to the best bound, depends dramatically on the problem. We now want to assume that the data errors do not exceed the data itself. For this we suppose

$$\|\mathbf{r}\|_{\ell_\infty(X)} \leq \delta \leq \|f\|_{W_2^\tau(\Omega)}$$

for a parameter $\delta > 0$.

Corollary 14 *If we choose the parameters as*

$$C = \frac{N\|f\|_{W_2^\tau(\Omega)}^2}{2\delta},$$
$$\varepsilon = \delta, \quad \textit{and} \quad \nu = \frac{1}{N},$$

we get

$$\left\|f - s_{X,\mathbf{y}}^{(\nu)}\right\|_{L_2(\Omega)} \leq \tilde{C}\left(h^\tau\|f\|_{W_2^\tau(\Omega)} + \delta\right)$$

and

$$\left\|f - s_{X,\mathbf{y}}^{(\nu)}\right\|_{L_\infty(\Omega)} \leq \tilde{C}\left(h^{\tau-d/2}\|f\|_{W_2^\tau(\Omega)} + \delta\right)$$

for all discrete sets $X \subset \Omega$ with fill distance $h := h_{X,\Omega} \leq C_\Omega \tau^{-2}$, with a generic positive constant $\tilde{C}$ which depends on τ, d and Ω but not on f or X.

8. ε-SVR with Exact Data

Since our arguments for the ν-SVR apply similarly to the ε-SVR, we skip over details and just state the results. Note that in this case the non-negative parameter ε is fixed in contrast to the free variable in the ν-SVR. Analogously to the notation introduced in the previous sections, we denote by $s_{X,\mathbf{y}}^{(\varepsilon)}$ the solution to the problem (10) for $X := \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\} \subset \Omega$ and $\mathbf{y} \in \mathbb{R}^N$. The stability and consistency estimates take the following form.

Lemma 15 *Under the assumption (11) concerning the data, we find that for every X and every fixed $\varepsilon \in \mathbb{R}^+$ a solution $s_{X,\mathbf{y}}^{(\varepsilon)}$ to problem (10) satisfies*

$$\left\|s_{X,\mathbf{y}}^{(\varepsilon)}\right\|_{\mathcal{N}_K(\Omega)} \leq \|f\|_{\mathcal{N}_K(\Omega)} \quad \textit{and}$$
$$\left\|s_{X,\mathbf{y}}^{(\varepsilon)}|_X - \mathbf{y}\right\|_{\ell_\infty(X)} \leq \frac{N}{2C}\|f\|_{\mathcal{N}_K(\Omega)}^2 + \varepsilon.$$

Again this leads to the following result on continuous L_q-norms.

Theorem 16 *We suppose $f \in W_2^\tau(\Omega)$ with $f\left(\mathbf{x}^{(i)}\right) = y_i$. Let $s_{X,\mathbf{y}}^{(\varepsilon)}$ be a solution of the ε-SVR, that is the optimization problem (10). Then there is a constant $\tilde{C} > 0$, which depends on τ, d and Ω but not on ε, f or X, such that the approximation error can be bounded by*

$$\left\| f - s_{X,\mathbf{y}}^{(\varepsilon)} \right\|_{L_q(\Omega)} \leq \tilde{C} \left(2h^{\tau - d\left(\frac{1}{2} - \frac{1}{q}\right)_+} \|f\|_{W_2^\tau(\Omega)} + \frac{N}{2C} \|f\|^2_{W_2^\tau(\Omega)} + \varepsilon \right) \tag{15}$$

for all discrete sets $X \subset \Omega$ with fill distance $h := h_{X,\Omega} \leq C_\Omega \tau^{-2}$.

Applying the same arguments as in the ν-SVR case we obtain the following corollary.

Corollary 17 *If we choose*

$$C = \frac{N \|f\|_{W_2^\tau(\Omega)}}{2h^\tau} \quad \text{, respectively } \; C = \frac{N \|f\|_{W_2^\tau(\Omega)}}{2h^{\tau - d/2}}$$

the inequality (15) turns into

$$\left\| f - s_{X,\mathbf{y}}^{(\varepsilon)} \right\|_{L_2(\Omega)} \leq \tilde{C} \left(3h^\tau \|f\|_{W_2^\tau(\Omega)} + \varepsilon \right) ,$$

respectively

$$\left\| f - s_{X,\mathbf{y}}^{(\varepsilon)} \right\|_{L_\infty(\Omega)} \leq \tilde{C} \left(3h^{\tau - \frac{d}{2}} \|f\|_{W_2^\tau(\Omega)} + \varepsilon \right)$$

for all discrete sets $X \subset \Omega$ with fill distance $h := h_{X,\Omega} \leq C_\Omega \tau^{-2}$, with a generic positive constant $\tilde{C}$ which depends on τ, d and Ω but not on $f \in W_2^\tau(\Omega)$ or X.

The rôle of the parameter C is similar to the one in case of the ν-SVR. But unlike in the case of the ν-SVR we are free to choose the parameter ε. We see that exact data implies that we should choose $\varepsilon \approx 0$. The case $C \to \infty$ and $\varepsilon \to 0$ leads to exact interpolation, and the well known error bounds for kernel-based interpolation (see Wendland, 2005) are attained.
We point out that the ε-SVR is closely related to the squared ε-loss,

$$\min_{s \in \mathcal{N}_K(\Omega)} \frac{1}{N} \sum_{j=1}^{N} \left| s\left(\mathbf{x}^{(j)}\right) - y_j \right|_\varepsilon^2 + \frac{1}{2C} \|s\|^2_{\mathcal{N}_K(\Omega)} \ . \tag{16}$$

This is important because for $\varepsilon = 0$ we get the square loss. Proceeding along the lines of this section, we find for a solution $s_{X,\mathbf{y}}^{(s\ell\varepsilon)}$ of (16) for exact data the stability bound

$$\left\| s_{X,\mathbf{y}}^{(s\ell\varepsilon)} \right\|_{\mathcal{N}_K(\Omega)} \leq \|f\|_{\mathcal{N}_K(\Omega)}$$

and the consistency estimate

$$\left\| s_{X,\mathbf{y}}^{(s\ell\varepsilon)}|_X - \mathbf{y} \right\|_{\ell_\infty(X)} \leq \sqrt{2} \left(\frac{N}{2C} \|f\|^2_{\mathcal{N}_K(\Omega)} + \varepsilon^2 \right)^{1/2} \leq \frac{\sqrt{N}}{\sqrt{C}} \|f\|_{\mathcal{N}_K(\Omega)} + \sqrt{2}\varepsilon \ .$$

Therefore, we obtain similar approximation results for the ε-squared loss as for the ε-SVR by inserting the estimates into the sampling inequalities. Similarly, the results of Section 9 can be adapted to the ε-squared loss. For the special case $\varepsilon = 0$, we obtain the usual least squares, which was analyzed by Wendland and Rieger (2005) in the case of exact data, and by Riplinger (2007) in the case of inexact data.

9. ε-SVR with Inexact Data

In this section we denote again by $s^{(\varepsilon)}_{X,\mathbf{y}}$ the solution to the problem (10) for a set of points $X := \{\mathbf{x}^{(1)},\dots,\mathbf{x}^{(N)}\} \subset \Omega$ and $\mathbf{y} \in \mathbb{R}^N$, but we allow the given data to be corrupted by some additive error according to assumption (14).

Lemma 18 *Under the assumption (14) concerning the data, for every X and every fixed $\varepsilon \in \mathbb{R}^+$ a solution $s^{(\varepsilon)}_{X,\mathbf{y}}$ to problem (10) satisfies*

$$\begin{aligned} \left\| s^{(\varepsilon)}_{X,\mathbf{y}} \right\|_{\mathcal{N}_K(\Omega)} &\leq \sqrt{\|f\|^2_{\mathcal{N}_K(\Omega)} + \frac{2C}{N}\sum_{i=1}^N |r_i|_\varepsilon} \quad \text{and} \\ \left\| s^{(\varepsilon)}_{X,\mathbf{y}}|_X - \mathbf{y} \right\|_{\ell_\infty(X)} &\leq \frac{N}{2C}\|f\|^2_{\mathcal{N}_K(\Omega)} + \sum_{i=1}^N |r_i|_\varepsilon + \varepsilon \,. \end{aligned}$$

These bounds shall now be plugged into the sampling inequality.

Theorem 19 *We suppose $f \in W_2^\tau(\Omega)$ with $f\left(\mathbf{x}^{(i)}\right) = y_i$. Let $s^{(\varepsilon)}_{X,\mathbf{y}}$ be a solution of the ε-SVR, that is the optimization problem (10). Then there is a constant $\tilde{C} > 0$, which depends on τ, d and Ω but not on ε, f or X, such that the approximation error can be bounded by*

$$\begin{aligned} \left\| f - s^{(\varepsilon)}_{X,\mathbf{y}} \right\|_{L_q(\Omega)} \leq\ & \tilde{C} \left(2h^{\tau - d\left(\frac{1}{2}-\frac{1}{q}\right)_+} \left(\|f\|_{W_2^\tau(\Omega)} + \sqrt{\|f\|^2_{W_2^\tau(\Omega)} + \frac{2C}{N}\sum_{i=1}^N |r_i|_\varepsilon} \right) \right. \\ & \left. + \frac{N}{2C}\|f\|^2_{W_2^\tau(\Omega)} + \sum_{i=1}^N |r_i|_\varepsilon + \varepsilon + \|\mathbf{r}\|_{\ell_\infty(X)} \right) \end{aligned}$$

for all discrete sets $X \subset \Omega$ with fill distance $h := h_{X,\Omega} \leq C_\Omega \tau^{-2}$.

If we again assume that the error level δ does not overrule the native space norm of the generating function,

$$\|\mathbf{r}\|_{\ell_\infty(X)} \leq \delta \leq \|f\|_{W_2^\tau(\Omega)} \,,$$

we get the following convergence orders, for our specific choices of the parameters.

Corollary 20 *Again we assume that the error satisfies (14). If we choose $\varepsilon = \delta$ and $C = \frac{N\|f\|_{W_2^\tau}}{2h^\tau}$ respectively $C = \frac{N\|f\|_{W_2^\tau}}{2h^{\tau-d/2}}$ then we find*

$$\begin{aligned} \left\| f - s^{(\varepsilon)}_{X,\mathbf{y}} \right\|_{L_2(\Omega)} &\leq \tilde{C}\left(h^\tau \|f\|_{W_2^\tau(\Omega)} + \delta \right) \quad \text{and} \\ \left\| f - s^{(\varepsilon)}_{X,\mathbf{y}} \right\|_{L_\infty(\Omega)} &\leq \tilde{C}\left(h^{\tau-d/2} \|f\|_{W_2^\tau(\Omega)} + \delta \right) \end{aligned}$$

for all discrete sets $X \subset \Omega$ with fill distance $h := h_{X,\Omega} \leq C_\Omega \tau^{-2}$, with a generic positive constant $\tilde{C}$ which depends on τ, d, and Ω but not on f or X.

10. Numerical Results

In this section we present some numerical examples to support our analytical results, in particular the rates of convergence in case of exact training data, and the detection of the error levels in case of noisy data.

10.1 Exact Training Data

Figure 1 illustrates the approximation orders in case of exact given data as considered in Sections 6 and 8. For that, we used regular data sets generated by the respective functions to be reconstructed and employed the ε- and the ν-SVR with the parameter choices provided in Corollaries 17 and 11, respectively. We implemented the finite dimensional formulations of the associated optimization problems as described in Equations (9) and (8). As kernel functions we used Wendland's functions for two reasons: On the one hand side they yield rather sparse kernel matrices $\mathbf{K}$ due to their compact support, on the other hand side they are easy to implement since they are piecewise polynomials. Furthermore Wendland's functions may be scaled to improve their numerical behaviour. An unscaled function K has support $\mathrm{supp}(K) \subset B(0,1) \subset \mathbb{R}^d$. The scaling is done in such a way that the decay of the Fourier transform is preserved, that is,

$$K^{(c)}(\mathbf{x}) = c^{-d} K\left(\frac{\mathbf{x}}{c}\right) , \quad \mathbf{x} \in \mathbb{R}^d . \tag{17}$$

By construction we have $supp\left(K^{(c)}\right) \subset B(0,c)$, such that small choices of the scaling parameter c imply rather sparse kernel matrices $\mathbf{K^{(c)}} = \left(K^{(c)}\left(\left\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\right\|_2\right)\right)_{i,j=1\ldots N}$. On the other hand side it is known that the constant factor in our error estimates increases with decreasing c. This is a typical trade-off situation between good approximation properties and good condition numbers of the kernel matrices $\mathbf{K^{(c)}}$ (Wendland, 2005). We chose a scaling $c = 0.1$ in all one-dimensional examples and a scaling $c = 2$ in all two-dimensional examples. Since these standard choices already work well, there was no need for a more careful choice. To our knowledge, there are so far no theoretical results on the optimal scaling.

The double logarithmic plots in Figure 1 visualize the convergence orders in terms of the fill distance. For that, the L_∞-approximation error $\|f - s_{X,\mathbf{y}}\|_{L_\infty}$ is plotted versus the fill distance h. The convergence rates can be found as the slopes of the lines.

In subfigure 1(a) the data was generated by

$$f(x) = (x - 0.5)_+^{2.5+eps} \in W_2^3([0,1]) ,$$

where *eps* denotes the relative machine precision in the sense of MATLAB. We use the notation $(t)_+ := \max\{0,t\}$ for all $t \in \mathbb{R}$. This function f is sampled on regular grids in the unit interval $I := [0,1]$ with 30 to 96 points. Note that in this case the fill distance is given by $h \approx 1/N$. We use two different kernel functions, namely (see Wendland, 2005)

- $K_1(x) = (1 - |x|)_+^3 (3|x| + 1)$ with native space $W_2^2([0,1])$, and
- $K_2(x) = (1 - |x|)_+^5 \left(8|x|^2 + 5|x| + 1\right)$ with native space $W_2^3([0,1])$.

The scaling parameter according to Equation (17) is chosen as $c = 0.1$. We employed the ε- and the ν- SVR with the parameter choices provided in Corollaries 17 and 11. The respective corollaries

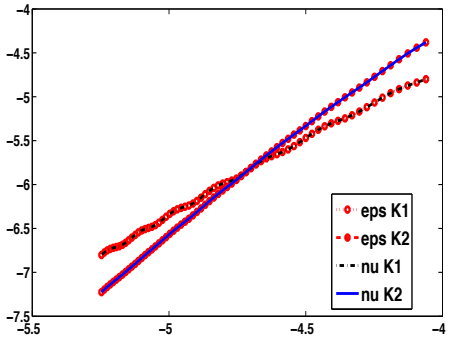
predict convergence rates of 1.5 for K_1, and 2.5 for K_2. In subfigure 1(a) the plots for the ε- and ν-SVR (almost identical) both show orders 1.7 for K_1 and 2.4 for K_2.
Subfigure 1(b) shows a 2-dimensional example. The data is generated by the smooth function

$$f(\mathbf{x}) = \sin(x_1 + x_2) \ .$$

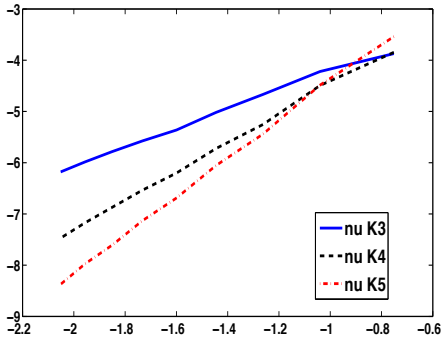
This function f is sampled on regular grids in the unit interval $I := [0,1]^2$ with 16 to 144 points. Note that in this case the fill distance is given by $h \approx \frac{1}{\sqrt{N}}$. We use three different kernel functions, namely (see Wendland, 2005)

- $K_3(\mathbf{x}) = (1 - \|\mathbf{x}\|)_+^4 (4\|\mathbf{x}\| + 1)$ with native space $W_2^{2.5}([0,1]^2)$,
- $K_4(\mathbf{x}) = (1 - \|\mathbf{x}\|)_+^6 \left(35\|\mathbf{x}\|^2 + 18\|\mathbf{x}\| + 3\right)$ with native space $W_2^{3.5}([0,1]^2)$, and
- $K_5(\mathbf{x}) = (1 - \|\mathbf{x}\|)_+^8 \left(32\|\mathbf{x}\|^3 + 25\|\mathbf{x}\|^2 + 8\|\mathbf{x}\| + 1\right)$ with native space $W_2^{4.5}([0,1]^2)$.

The kernel functions were scaled by $c = 2$ according to Equation (17). For the sake of simplicity we employed only the ν-SVR with the parameter choices provided in Corollary 11. The predicted convergence rates in the fill distance h are 1.5 for K_3, 2.5 for K_4 and 3.5 for K_5. The numerical experiments show orders 1.8 for K_3, 2.8 for K_4 and 3.7 for K_5. Therefore, the numerical examples support our analytical results.



(a) Data generated by $f \in W_2^3(I)$ on regular grids in I. ν- and ε-SVR yield orders 1.7 for K_1, and 2.4 for K_2. Scaling parameter $c = 0.1$.



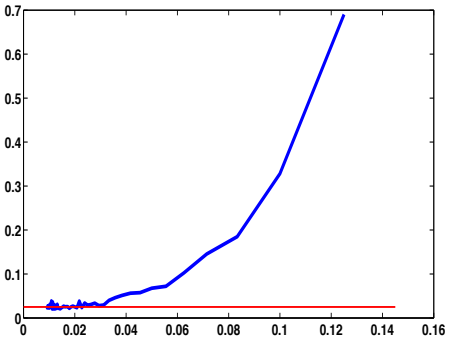
(b) Data generated by smooth function on regular grids in I^2. ν-SVR yields orders 1.8 for K_3, 2.8 for K_4, and 3.7 for K_5. Scaling parameter $c = 2$.

Figure 1: Logarithm of the L_∞-approximation error plotted versus the logarithm of the fill distance h for exact training data.

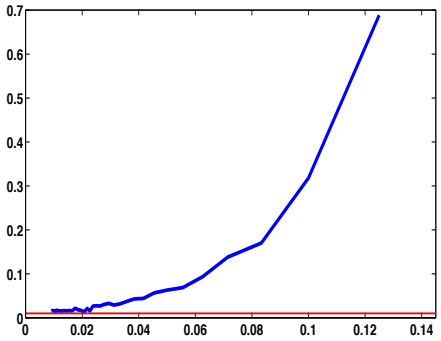
10.2 Inexact Data

Figure 2 shows examples for the case of noisy data. The plots show the L_∞-approximation error $\|f - s_{X,\mathbf{y}}\|_{L_\infty}$ versus the fill distance h. For simplicity we concentrated on the case of the ν-SVR in the one dimensional setting. We used the noise model given by Equation (14), that is $y = f + r$. In Subfigure 2(a) the function $f(x) = \sin(10x)$ is sampled on regular grids of 2 to 56 points in $[0,1]$. The data is disturbed by an error r which is normally distributed with mean zero and standard deviation 0.01. As kernel function we use K_1, and the parameters of the ν-SVR are chosen as in

Corollary 14. The plot shows that for $h \to 0$ the error remains of the same order of magnitude as the error level $\|r\|_{\ell_\infty}$.
In Subfigure 2(b) the function $f(x) = \sin(10x)$ is sampled on regular grids of 5 to 56 points in the unit interval $I = [0,1]$. Here, the data is corrupted by an error of ± 0.01, where the sign of the error is chosen randomly with equal likelihood for plus and minus. As kernel function we use K_1 with $c = 0.3$, and the parameters of the ν-SVR are chosen as in Corollary 14. The plot shows that the L_∞-approximation error converges to a constant of the order of magnitude of the error level for $h \to 0$.



(a) Data disturbed by random error with mean zero and standard deviation 0.01. Approximation error for $h \to 0$ reaches the error level and remains bounded of the same order of magnitude as the error level.



(b) Data disturbed by random sign deterministic error ± 0.01. Approximation error converges to a constant of the order of magnitude of the error level for $h \to 0$.

Figure 2: L_∞-approximation error versus fill distance in case of inexact data.

11. Summary and Outlook

We proved deterministic worst-case error estimates for kernel-based regression algorithms. The main ingredient are sampling inequalities. We provided a detailed analysis only for the ν- and the ε-SVR for both exact and inexact training data. However, the same techniques apply to all machine learning problems involving penalty terms induced by kernels related to Sobolev spaces. If the function to be reconstructed lies in the reproducing kernel Hilbert space (RKHS) of an infinitely smooth kernel such as the Gaussian or an infinite dot product kernel, a similar analysis based on sampling inequalities can be done, leading to exponential convergence rates (see Rieger and Zwicknagl 2008 and Zwicknagl 2009 for first results in this direction).
So far, our error estimates depend explicitly on the pointwise noise in the data, and we do not make any assumptions on the noise distribution. Future work should incorporate probabilistic models on the noise distribution to yield estimates for the expected error.

Acknowledgments

We thank Professor Robert Schaback for helpful discussions and his continued support. Further thanks go to the referees for several valuable comments. CR was supported by the Deutsche Forschungsgemeinschaft through the Graduiertenkolleg 1023 *Identification in Mathematical Models: Synergy of Stochastic and Numerical Methods*. BZ would like to thank the German National Academic Foundation (Studienstiftung des deutschen Volkes) for their support.

References

S.C. Brenner and L.R. Scott. *The Mathematical Theory of Finite Element Methods*, volume 15 of *Texts in Applied Mathematics*. Springer, New York, 1994.

C-C. Chang and C-L. Lin. Training ν-support vector regression: Theory and algorithms. *Neural Computation*, 14(8):1959–1977, 2002.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10 (8):1455–1480, 1998.

C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

C. Rieger and B. Zwicknagl. Sampling inequalities for infinitely smooth functions, with applications to interpolation and machine learning. To appear in *Advances in Computational Mathematics*, 2008.

M. Riplinger. Lernen als inverses Problem und deterministische Fehlerabschätzung bei Support Vektor Regression. Diplomarbeit, Universität des Saarlandes, 2007.

R. Schaback. The missing wendland functions. To appear in *Advances in Computational Mathematics*, 2009.

B. Schölkopf and A.J. Smola. *Learning with kernels - Support Vector Machines, Regularisation, and Beyond*. MIT Press, Cambridge, Massachusetts, 2002.

B. Schölkopf, C. Burges, and V.Vapnik. Extracting support data for a given task. In *Proceedings, First International Conference on Knowledge Discovery and Data Mining*. CA:AAAI Press., Menlo Park, 1995.

B. Schölkopf, R.C. Wiliamson, and P.L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

C.J. Stone. Optimal global rates of convergence for nonparametric regression. *The Annals of Statistics*, 10:1040–1053, 1982.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

H. Wendland. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, 2005.

H. Wendland and C. Rieger. Approximate interpolation. *Numerische Mathematik*, 101:643–662, 2005.

J. Wloka. *Partielle Differentialgleichungen: Sobolevräume und Randwertaufgaben*. Mathematische Leitfäden. Teubner, Stuttgart, 1982.

B. Zwicknagl. Power series kernels. *Constructive Approximation*, 29(1):61–84, 2009.

Journal of Machine Learning Research 10 (2009) 2133-2136 Submitted 3/09; Published 9/09

RL-Glue: Language-Independent Software for Reinforcement-Learning Experiments

Brian Tanner BTANNER@CS.UALBERTA.CA
Adam White AWHITE@CS.UALBERTA.CA
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8

Editor: Mikio L. Braun

Abstract

RL-Glue is a standard, language-independent software package for reinforcement-learning experiments. The standardization provided by RL-Glue facilitates code sharing and collaboration. Code sharing reduces the need to re-engineer tasks and experimental apparatus, both common barriers to comparatively evaluating new ideas in the context of the literature. Our software features a minimalist interface and works with several languages and computing platforms. RL-Glue compatibility can be extended to any programming language that supports network socket communication. RL-Glue has been used to teach classes, to run international competitions, and is currently used by several other open-source software and hardware projects.

Keywords: reinforcement learning, empirical evaluation, standardization, open source

1. Introduction and Motivation

Reinforcement learning is an embodied, trial-and-error problem formulation for artificial intelligence (Sutton and Barto, 1998; Kaelbling et al., 1996; Bertsekas and Tsitsiklis, 1996). At a series of time steps, the *agent* emits actions in response to observations and rewards generated by the *environment*. The agent's objective is to select actions that maximize the future rewards. Reinforcement-learning methods have been successfully applied to many problems including backgammon (Tesauro, 1994), elevator control (Crites and Barto, 1998), and helicopter control (Ng et al., 2004). Reinforcement-learning models and formalisms have influenced a number of fields, including operations research, cognitive science, optimal control, psychology, neuroscience, and others.

Reinforcement-learning practitioners create their agents and environments using various incompatible software frameworks, making collaboration inconvenient and thus slowing progress in our community. It can be time consuming, difficult, and sometimes even impossible to exactly reproduce the work of others. A conference or journal article is not the appropriate medium to share a sufficiently detailed specification of the environment, agent and overall experimental apparatus. We need a convenient way to share source code.

We believe that a standard programming interface for reinforcement-learning experiments will remove some barriers to collaboration and accelerate the pace of research in our field. To encourage widespread adoption, this interface should be easy to adhere to, and it should not force users to abandon their favorite tools or languages. With these goals in mind, we have developed RL-Glue: language independent software for reinforcement-learning experiments.

2. RL-Glue

Reinforcement-learning environments cannot be stored as fixed data-sets, as is common in conventional supervised machine learning. The environment generates observations and rewards in response to actions selected by the agent, making it more natural to think of the environment and agent as interactive *programs*. Sutton and Barto describe one prevalent view of agent-environment interactions in their introductory text (1998). Their view, shown in Figure 1, clearly separates the agent and environment into different components which interact in a particular way, following a particular sequence.

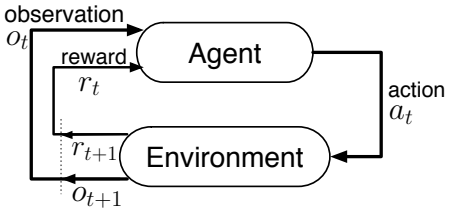


Figure 1: Sutton and Barto's agent-environment interface, with states generalized to observations.

White's RL-Glue Protocol (2006) formalizes Sutton and Barto's interface for online, single-agent reinforcement learning. The RL-Glue Protocol describes how the different aspects of a reinforcement-learning experiment should be arranged into programs, and the etiquette they should follow when communicating with each other. These programs (Figure 2) are the agent, the environment, the experiment, and RL-Glue. The agent program implements the learning algorithm and action-selection mechanism. The environment program implements the dynamics of the task and generates the observations and rewards. The experiment program directs the experiment's execution, including the sequence of agent-environment interactions and agent performance evaluation. The RL-Glue program mediates the communication between the agent and environment programs in response to commands from the experiment program. Our RL-Glue software (RL-Glue) implements White's protocol.[1]

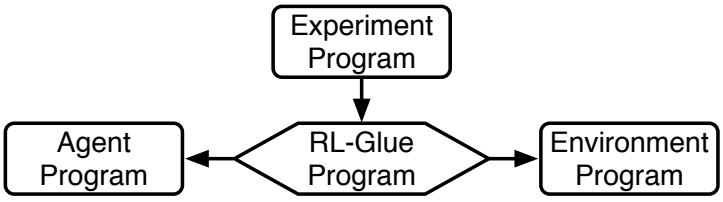


Figure 2: The four programs specified by the RL-Glue Protocol. Arrows indicate the direction of the flow of control.

RL-Glue can be used either in internal or external mode. In *internal* mode, the agent, environment and experiment are linked into a single program, and their communication is through function calls. Internal mode is currently an option if the agent, environment, and experiment are written exclusively in Java or C/C++. In *external* mode, the agent, environment and experiment are linked

1. This can be found at `http://glue.rl-community.org/protocol`.

into separate programs. Each program connects to the RL-Glue server program, and all communication is over TCP/IP socket connections. External mode allows these programs to be written in any programming language that supports socket communication. External mode is currently supported for C/C++, Java, Python, Lisp, and Matlab.

Each mode has its strengths and weaknesses. Internal mode has less overhead, so it can execute more steps per second. External mode is more flexible and portable. The performance difference between the two modes vanishes as the agent or environment becomes complex enough that computation dominates the socket overhead in terms of time per step. The agent and environment are indifferent and unaware of their execution mode; the difference in modes lies only in how the agent and environment are linked or loaded.

3. RL-Glue in Practice

RL-Glue's provides a common interface for a number of software and hardware projects in the reinforcement-learning community. For example, there is the annual RL-Competition, where teams from around the world compare their agents on a variety of challenging environments. The competition software uses the API, called RL-Viz, that is layered on top of RL-Glue to dynamically load agent and environment programs, modify parameters at runtime and visualize interaction and performance. All of the environments and sample agents created by the competition organizers are added to the RL-Library, a public, community-supported repository of RL-Glue compatible code. The RL-Library is also available as an archive of top competition agents, challenge problems, project code from academic publications, or any other RL-Glue compatible software that members of our community would like to share.

The socket architecture of RL-Glue allows diverse software and hardware platforms to be connected as RL-Glue environment programs. There are ongoing projects that connect a mobile robot platform, a keepaway soccer server, a real-time strategy game, and an Atari emulator to RL-Glue. Our socket architecture helps lower the barriers for researchers wishing to work on larger scale environments by providing a simple and familiar interface.

RL-Glue has been used for teaching reinforcement learning in several university courses and to create experiments for scientific articles published in leading conferences. See our *RL-Glue in practice* web page for an updated list of projects and papers that have used RL-Glue.[2]

4. Other Reinforcement-Learning Software Projects

RL-Glue is not the first software project that aims to standardize empirical reinforcement learning or to make agent and environment programs more accessible within our community. However, RL-Glue is the only project that offers a standardized language-independent interface, rich actions and observations, and fine-grained control of the experiment.

Other projects, most notably: CLSquare,[3] PIQLE,[4] RL Toolbox,[5] JRLF,[6] and LibPG,[7] offer significant value to the reinforcement-learning community by offering agents and environments,

2. This can be found at `http://glue.rl-community.org/rl-glue-in-practice`.
3. This can be found at `http://www.ni.uos.de/index.php?id=70`.
4. This can be found at `http://piqle.sourceforge.net/`.
5. This can be found at `http://www.igi.tugraz.at/ril-toolbox/`.
6. This can be found at `http://mykel.kochenderfer.com/jrlf/`.
7. This can be found at `http://code.google.com/p/libpgrl/`.

intuitive visualizations, programming tools, etc. Users should not be forced to choose between RL-Glue and these alternative projects. Our design makes it relatively easy to interface existing frameworks with RL-Glue. We are currently offering our assistance in bridging other frameworks to RL-Glue, with the hope of improving access to all of these tools for all members of our community.

5. RL-Glue Open Source Project

Website: http://glue.rl-community.org **License:** `Apache 2.0`

RL-Glue is more than an interface; it connects a family of community projects, with many levels of possible participation. Members of the community are invited to submit agent, environment and experiment programs to the RL-Library. Developers can also extend the reach of RL-Glue compatibility by writing external-mode or internal-mode interfaces for their favorite programming language. The RL-Glue software project also welcomes submissions and improvements for all parts of the software and documentation.

Acknowledgments

We would like to thank the users, testers, and developers for their contributions to RL-Glue 3.0. Special thanks to Gábor Balázs, José Antonio Martin H., Scott Livingston, Marc Bellemare, István Szita, Marc Lanctot, Anna Koop, Dan Lizotte, Richard Sutton, Monica Dinculescu, Jordan Frank, and Andrew Butcher. Of course, we also owe a great debt to all of the talented people responsible for the historic and ongoing development of RL-Glue.[8]

References

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, May 1996. ISBN 1886529108.

Robert H. Crites and Andrew G. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2-3):235–262, 1998.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Proceedings of the International Symposium on Experimental Robotics*, pages 363–372, 2004.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts, 1998.

Gerald Tesauro. TD-gammon, a self-teaching backgammon program achieves master-level play. *Neural Computation*, 6:215–219, 1994.

Adam White. A Standard System for Benchmarking in Reinforcement Learning. Master's thesis, University of Alberta, Alberta, Canada, 2006.

8. This can be found at `http://glue.rl-community.org/contributors-history`.

Journal of Machine Learning Research 10 (2009) 2137-2155 Submitted 4/08; Revised 7/09; Published 9/09

Discriminative Learning Under Covariate Shift

Steffen Bickel BICKEL@CS.UNI-POTSDAM.DE
Michael Brückner MIBRUECK@CS.UNI-POTSDAM.DE
Tobias Scheffer SCHEFFER@CS.UNI-POTSDAM.DE
University of Potsdam, Department of Computer Science
August-Bebel-Str. 89
14482 Potsdam, Germany

Editor: Bianca Zadrozny

Abstract

We address classification problems for which the training instances are governed by an input distribution that is allowed to differ arbitrarily from the test distribution—problems also referred to as classification under covariate shift. We derive a solution that is purely discriminative: neither training nor test distribution are modeled explicitly. The problem of learning under covariate shift can be written as an integrated optimization problem. Instantiating the general optimization problem leads to a kernel logistic regression and an exponential model classifier for covariate shift. The optimization problem is convex under certain conditions; our findings also clarify the relationship to the known kernel mean matching procedure. We report on experiments on problems of spam filtering, text classification, and landmine detection.

Keywords: covariate shift, discriminative learning, transfer learning

1. Introduction

Most machine learning algorithms are constructed under the assumption that the training data is governed by the exact same distribution which the model will later be exposed to. In practice, control over the data generation process is often less perfect. Training data may be obtained under laboratory conditions that cannot be expected after deployment of a system; spam filters may be used by individuals whose distribution of inbound emails diverges from the distribution reflected in public training corpora; image processing systems may be deployed to foreign geographic regions where vegetation and lighting conditions result in a distinct distribution of input patterns.

The case of distinct training and test distributions in a learning problem has been referred to as *covariate shift* and *sample selection bias*—albeit the term sample selection bias actually refers to a case in which each training instance is originally drawn from the test distribution, but is then selected into the training sample with some probability, or discarded otherwise.

The covariate shift model and the *missing at random* case in the sample selection bias model allow for differences between the training and test distribution of instances; the conditional distribution of the class variable given the instance is constant over training and test set.

In the *covariate shift* problem setting, a training sample is available in matrix $\mathbf{X}_L$ with row vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$. This training sample is governed by an unknown distribution $p(\mathbf{x}|\lambda)$. Vector $\mathbf{y}$ with elements $y_1, \ldots, y_m$ are the labels for training examples and are drawn according to an unknown target concept $p(y|\mathbf{x})$. In addition, unlabeled test data becomes available in matrix $\mathbf{X}_T$ with rows

$\mathbf{x}_{m+1}, \ldots, \mathbf{x}_{m+n}$. The test data is governed by a different unknown distribution, $p(\mathbf{x}|\theta)$. Training and test distribution may differ arbitrarily, but there is only one unknown target conditional class distribution $p(y|\mathbf{x})$.

In discriminative learning tasks such as classification, the classifier's goal is to produce the correct output given the input. It is widely accepted that this is best performed by discriminative learners that directly maximize a quality measure of the produced output. Model-based optimization criteria such as the joint likelihood of input and output, by contrast, additionally assess how well the classifier models the distribution of input values. This amounts to adding a term to the criterion that is irrelevant for the task at hand.

We contribute a discriminative model for learning under different training and test distributions. The model directly characterizes the divergence between training and test distribution, without the intermediate—intrinsically model-based—step of estimating training and test distribution. We formulate the search for all model parameters as an integrated optimization problem. This complements the predominant procedure of first estimating the bias of the training sample, and then learning the classifier on a weighted version of the training sample. We show that the integrated optimization can be convex, depending on the model type; it is convex for the exponential model. We derive a Newton gradient descent procedure, leading to a kernel logistic regression and an exponential model classifier for covariate shift.

After reviewing models for differing training and test distributions in Section 2, we introduce our integrated model in Section 3. We derive primal and kernelized classifiers for differing training and test distributions in Sections 4 and 5. In Section 6, we analyze the convexity of the integrated optimization problem. Section 7 describes an approximation to the joint optimization problem and Section 8 reveals a new interpretation of kernel mean matching and analyzes the relationship to our model. In Section 9 we discuss different tuning procedures for learning under covariate shift. Section 10 provides empirical results and Section 11 concludes.

The discriminative model for the logistic loss is described in a prior conference publication (Bickel et al., 2007). Our original results showed that the resulting optimization problem is not convex. New findings (Section 6) show that the integrated optimization problem can in fact be convex when the loss function is chosen appropriately. Section 7 describes a two-stage approximation that allows to train virtually any type of classifier under covariate shift. The new Section 8 characterizes the relation to kernel mean matching. New experiments include the exponential target model. Section 10 uses an experimental setting that differs from the setting of Bickel et al. (2007) in the parameter tuning process. In some cases, the new setting has improved the performance of baseline methods.

2. Prior Work

If training and test distributions were known, then the loss on the test distribution could be minimized by weighting the loss on the training distribution with an instance-specific factor. Proposition 1 (Shimodaira, 2000) illustrates that the scaling factor has to be $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$.

Proposition 1 *The expected loss with respect to* θ *equals the expected loss with respect to* λ *with weights* $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ *for the loss incurred by each* $\mathbf{x}$*, provided that the support of* $p(\mathbf{x}|\theta)$ *is contained in the*

support of $p(\mathbf{x}|\lambda)$*:*

$$\mathbf{E}_{(\mathbf{x},y)\sim\theta}[\ell(f(\mathbf{x}),y)] \quad = \quad \mathbf{E}_{(\mathbf{x},y)\sim\lambda}\left[\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}\ell(f(\mathbf{x}),y)\right]. \tag{1}$$

After expanding the expected value into its integral $\int \ell(f(\mathbf{x}),y)p(\mathbf{x},y|\theta)d\theta$, the joint distribution $p(\mathbf{x},y|\lambda)$ is decomposed into $p(\mathbf{x}|\lambda)p(y|\mathbf{x},\lambda)$. Since $p(y|\mathbf{x},\lambda) = p(y|\mathbf{x}) = p(y|\mathbf{x},\theta)$ is the global conditional distribution of the class variable given the instance, Proposition 1 follows. All instances $\mathbf{x}$ with positive $p(\mathbf{x}|\theta)$ are integrated over. Hence, Equation 1 holds as long as each $\mathbf{x}$ with positive $p(\mathbf{x}|\theta)$ also has a positive $p(\mathbf{x}|\lambda)$; otherwise, the denominator vanishes. This shows that covariate shift can only be compensated for as long as the training distribution covers the entire support of the test distribution. If a test instance had zero density under the training distribution, the test-to-training density ratio which it would need to be scaled with would incur a zero denominator.

Both, $p(\mathbf{x}|\theta)$ and $p(\mathbf{x}|\lambda)$ are unknown, but $p(\mathbf{x}|\theta)$ is reflected in $\mathbf{X}_T$, as is $p(\mathbf{x}|\lambda)$ in $\mathbf{X}_L$. A straightforward approach to compensating for covariate shift is to first obtain estimates $\hat{p}(\mathbf{x}|\theta)$ and $\hat{p}(\mathbf{x}|\lambda)$ from the test and training data, respectively, using kernel density estimation (Shimodaira, 2000; Sugiyama and Müller, 2005). In a second step, the estimated density ratio is used to re-sample the training instances, or to train with weighted examples.

This method decouples the problem. First, it estimates training and test distributions. This step is intrinsically model-based and only loosely related to the ultimate goal of accurate classification. In a subsequent step, the classifier is derived given fixed weights. Since the parameters of the final classifier and the parameters that control the weights are not independent, this decomposition into two optimization steps cannot generally find the optimal setting of the *joint* parameter vector.

A line of work on learning under sample selection bias has meandered from the statistics and econometrics community into machine learning (Heckman, 1979; Zadrozny, 2004). Sample selection bias relies on a model of the data generation process. Test instances are drawn under $p(\mathbf{x}|\theta)$. Training instances are drawn by first sampling $\mathbf{x}$ from the test distribution $p(\mathbf{x}|\theta)$. A selector variable σ then decides whether $\mathbf{x}$ is moved into the training set ($\sigma = 1$) or moved into the rejected set ($\sigma = -1$). For instances in the training set ($\sigma = 1$) a label is drawn from $p(y|\mathbf{x})$, for the instances in the rejected set the labels are unknown. A typical scenario for sample selection bias is credit scoring. The labeled training sample consists of customers who where given a loan in the past and the rejected sample are customers that asked for but where not given a loan. New customers asking for a loan reflect the test distribution.

In the *missing at random* case, the selector variable is only dependent on $\mathbf{x}$, but not on y; that is, $p(\sigma = 1|\mathbf{x},y,\theta,\lambda) = p(\sigma = 1|\mathbf{x},\theta,\lambda)$. The distribution of the selector variable then maps the test onto the training distribution:

$$p(\mathbf{x}|\lambda) \propto p(\mathbf{x}|\theta)p(\sigma = 1|\mathbf{x},\theta,\lambda).$$

Proposition 2 (Zadrozny, 2004; Bickel and Scheffer, 2007) says that minimizing the loss on instances weighted by $p(\sigma|\mathbf{x},\theta,\lambda)^{-1}$ in fact minimizes the expected loss with respect to θ.

Proposition 2 *The expected loss with respect to* θ *is proportional to the expected loss with respect to* λ *with weights* $p(\sigma = 1|\mathbf{x},\theta,\lambda)^{-1}$ *for the loss incurred by each* $\mathbf{x}$*, provided that the support of* $p(\mathbf{x}|\theta)$ *is contained in the support of* $p(\mathbf{x}|\lambda)$*:*

$$\mathbf{E}_{(\mathbf{x},y)\sim\theta}[\ell(f(\mathbf{x}),y)] \quad \propto \quad \mathbf{E}_{(\mathbf{x},y)\sim\lambda}\left[\frac{1}{p(\sigma = 1|\mathbf{x},\theta,\lambda)}\ell(f(\mathbf{x}),y)\right].$$

When the model is implemented, $p(\sigma = 1|\mathbf{x},\theta,\lambda)$ is learned by discriminating the training against the rejected examples; in a second step the target model is learned by following Proposition 2 and weighting training examples by $p(\sigma|\mathbf{x},\theta,\lambda)^{-1}$. No test examples drawn directly from $p(\mathbf{x}|\theta)$ are needed to train the model, only labeled selected and unlabeled rejected examples are required. This is in contrast to the covariate shift model that requires samples drawn from the test distribution, but no selection process is assumed and no rejected examples are needed. Covariate shift models can be applied to learning under sample selection bias in the missing at random setting by treating the selected examples as labeled sample and the union of selected (ignoring the labels) and rejected examples as unlabeled sample.

Propensity scores (Rosenbaum and Rubin, 1983; Lunceford and Davidian, 2004) are applied in settings related to sample selection bias; the training data is again assumed to be drawn from the test distribution $p(\mathbf{x}|\theta)$ followed by a selection process. The difference to the setting of sample selection bias is that the selected *and* the rejected examples are labeled. Weighting the selected examples by the inverse of the propensity score $p(\sigma = 1|\mathbf{x},\lambda,\theta)^{-1}$ and weighting the rejected examples by $p(\sigma = -1|\mathbf{x},\lambda,\theta)^{-1}$ results in two unbiased samples with respect to the test distribution.

Propensity scoring can precede a variety of analysis steps. This can be the training of a target model on re-weighted data or just a statistical analysis of the two re-weighted samples. A typical application for propensity scores is the analysis of the success of a medical treatment. Patients are selected to be given the treatment and some other patients are selected into the control group. If the selector variable is not independent of $\mathbf{x}$ (patients may be chosen for an experimental therapy only if they meet specific requirements), the outcome (e.g., ratio of cured patients) of the two groups cannot be compared directly, propensity scores have to be applied.

Maximum entropy density estimation under sample selection bias has been studied by Dudik et al. (2005). Bickel and Scheffer (2007) impose a Dirichlet process prior on several learning problems with related sample selection bias. Elkan (2001) and Japkowicz and Stephen (2002) investigate the case of training data that is only biased with respect to the class ratio, this can be seen as sample selection bias where the selection only depends on y.

Kernel mean matching (Huang et al., 2007) is a two-step method that first finds weights for the training instances such that the first momentum of training and test sets—that is, their mean value—matches in feature space. The subsequent training step uses these weights. Matching the means in feature space is equivalent to matching all moments of the distributions if a universal kernel is used. Huang et al. (2007) derive a quadratic program from Equation 2 that can be solved with standard optimization tools. $\Phi(\cdot)$ is a mapping into a feature space and B is a regularization parameter.

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \left\| \tfrac{1}{m}\sum_{i=1}^{m} \alpha_i \Phi(\mathbf{x}_i) - \tfrac{1}{n}\sum_{i=m+1}^{m+n} \Phi(\mathbf{x}_i) \right\|^2 \\ &\text{subject to } \alpha_i \in [0,B] \text{ and } \left| \tfrac{1}{m}\sum_{i=1}^{m} \alpha_i - 1 \right| \leq \varepsilon \end{aligned} \tag{2}$$

Cortes et al. (2008) theoretically analyze the error that gets introduced by estimating sample selection bias from data. Their analysis covers the kernel mean matching procedure and a cluster-based estimation technique.

KLIEP (Sugiyama et al., 2008) estimates resampling weights for the training examples by minimizing the Kullback-Leibler divergence between the test distribution and the weighted training distribution. Tsuboi et al. (2008) derive an extension to KLIEP for large-scale applications and reveal a close relationship to kernel mean matching.

3. Integrated Model

Our goal is to find model parameters $\mathbf{w}$ for a probabilistic classification model $f(\mathbf{x}) = \text{argmax}_y\, p(y|\mathbf{x};\mathbf{w})$. The model should correctly predict labels of the test data $\mathbf{X}_T$ drawn from $p(\mathbf{x}|\theta)$. A regular *maximum a posteriori* estimation $\mathbf{w}' = \text{argmax}_\mathbf{w}\, p(\mathbf{y}|\mathbf{X}_L;\mathbf{w})p(\mathbf{w})$, would only use the training data $(\mathbf{y},\mathbf{X}_L)$ governed by $p(\mathbf{x}|\lambda)$. By ignoring the test data, this estimate will not generally result in a model that predicts the missing labels of the test data with a minimum error because the training distribution $p(\mathbf{x}|\lambda)$ is different from the test distribution $p(\mathbf{x}|\theta)$.

In the following we devise a probabilistic model that accounts for the difference between training and test distribution. Before we describe the model we define a joint data matrix $\mathbf{X}$ that is a concatenation of the matrices $\mathbf{X}_L$ and $\mathbf{X}_T$. The model is based on a binary selector variable s: Given an instance vector $\mathbf{x}$ from the joint matrix $\mathbf{X}$ of all available instances, selector variable s decides whether $\mathbf{x}$ is drawn into the test data $\mathbf{X}_T$ ($s=-1$) or into the training data $\mathbf{X}_L$ ($s=1$) in which case $\mathbf{y}$ is determined. The variable s is governed by the distribution $p(s|\mathbf{x};\mathbf{v})$. Parameter $\mathbf{v}$ characterizes the discrepancy between the training and test distribution. Based on the model for s we can now describe the generative process underlying our model:

1. Draw parameter vectors $\mathbf{v}$ and $\mathbf{w}$ from prior distributions $p(\mathbf{v})$ and $p(\mathbf{w})$;
2. For each row $\mathbf{x}$ in matrix $\mathbf{X}$ draw binary variable s from distribution $p(s|\mathbf{x};\mathbf{v})$; accordingly, the likelihood of the vector of all selector variables $\mathbf{s}$ is $p(\mathbf{s}|\mathbf{X};\mathbf{v}) = \prod_{i=1}^{m+n} p(s_i|\mathbf{x}_i;\mathbf{v})$;
3. For all selected training examples (all examples $\mathbf{x}_i$ with $s_i = 1$) draw vector $\mathbf{y}$ of all labels from $p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})$.

This generative process corresponds to the following factorization of the joint probability of the vector of labels $\mathbf{y}$, vector of selector variables $\mathbf{s}$, and parameter vectors $\mathbf{v}$ and $\mathbf{w}$:

$$p(\mathbf{y},\mathbf{s},\mathbf{w},\mathbf{v}|\mathbf{X}) \;=\; p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})p(\mathbf{s}|\mathbf{X};\mathbf{v})p(\mathbf{w})p(\mathbf{v}). \tag{3}$$

3.1 Maximum A Posteriori Parameter Inference

For parameter inference we want to find parameters $\mathbf{w}$ that maximize the posterior probability given all available data (Equation 4). The available data are the data matrix $\mathbf{X}$, the label vector $\mathbf{y}$, and the selection vector $\mathbf{s}$, that splits the data matrix into training and test data. Because the parameter $\mathbf{v}$ is unknown and is not needed for the final classifier the best we can do is to integrate it out (Equation 5).

$$\mathbf{w}^* \;=\; \text{argmax}_\mathbf{w}\, p(\mathbf{w}|\mathbf{y},\mathbf{s},\mathbf{X}) \tag{4}$$

$$\;=\; \text{argmax}_\mathbf{w} \int p(\mathbf{w},\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X})d\mathbf{v}. \tag{5}$$

Integrating over $\mathbf{v}$ is computationally infeasible. In Equation 6, the integral is therefore approximated by the single assignment of values to the parameters which maximizes the posterior—the *maximum a posteriori* (MAP) estimator. In our case, the MAP estimator naturally assigns values to all parameters, $\mathbf{w}$ and $\mathbf{v}$.

$$(\mathbf{w}_{\text{MAP}},\mathbf{v}_{\text{MAP}}) \;=\; \text{argmax}_{\mathbf{w},\mathbf{v}}\, p(\mathbf{w},\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X}) \tag{6}$$

$$\;=\; \text{argmax}_{\mathbf{w},\mathbf{v}}\, p(\mathbf{y},\mathbf{s},\mathbf{w},\mathbf{v}|\mathbf{X}) \tag{7}$$

$$\;=\; \text{argmax}_{\mathbf{w},\mathbf{v}}\, p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})p(\mathbf{s}|\mathbf{X};\mathbf{v})p(\mathbf{w})p(\mathbf{v}). \tag{8}$$

Equation 7 follows from multiplication with a constant $p(\mathbf{y},\mathbf{s}|\mathbf{X})$ and from application of the chain rule. Equation 8 applies the factorization from the generative process of Equation 3.

The class-label posterior $p(y|\mathbf{x};\mathbf{w}_{\text{MAP}})$ is conditionally independent of $\mathbf{v}_{\text{MAP}}$ given $\mathbf{w}_{\text{MAP}}$. However, $\mathbf{w}_{\text{MAP}}$ and $\mathbf{v}_{\text{MAP}}$ are dependent. Assigning a single MAP value to $[\mathbf{w},\mathbf{v}]$ instead of integrating over $\mathbf{v}$ is a common approximation. However, sequential maximization of $p(\mathbf{s}|\mathbf{X};\mathbf{v})$ over parameters $\mathbf{v}$ followed by maximization of $p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})$ over parameters $\mathbf{w}$ with fixed $\mathbf{v}$ would amount to an additional degree of approximation and will not generally coincide with the maximum of the product in Equation 8. Such a sequential maximization corresponds to the predominant two-step procedure for learning under covariate shift.

In the next sections we will discuss the likelihood functions $p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})$ and $p(\mathbf{s}|\mathbf{X};\mathbf{v})$ and the optimization problem for parameter inference based on maximization of Equation 8.

3.2 Label Likelihood and Discriminative Weighting Factors

In order to define the label likelihood we first derive a discriminative expression for $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ which will no longer include any density on instances. When $p(s=-1)>0$, which is implied by the test set not being empty, the definition of s allows us to rewrite the test distribution as $p(\mathbf{x}|\theta)=p(\mathbf{x}|s=-1,\theta)$. Since test instances are only dependent on parameter θ but not on parameter λ, equation $p(\mathbf{x}|s=-1,\theta)=p(\mathbf{x}|s=-1,\theta,\lambda)$ follows. By an analogous argument, $p(\mathbf{x}|\lambda)=p(\mathbf{x}|s=1,\theta,\lambda)$ when $p(s=1)>0$. This implies Equation 9.

In Equation 10, Bayes' rule is applied twice; the two terms of $p(\mathbf{x}|\theta,\lambda)$ cancel each other out in Equation 11. Since $p(s=-1|\mathbf{x},\theta,\lambda)=1-p(s=1|\mathbf{x},\theta,\lambda)$, Equation 12 follows.

The conditional $p(s=1|\mathbf{x},\theta,\lambda)$ discriminates training ($s=1$) against test instances ($s=-1$).

$$\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)} = p(\mathbf{x}|s=-1,\theta,\lambda)\frac{1}{p(\mathbf{x}|s=1,\theta,\lambda)} \tag{9}$$

$$= \frac{p(s=-1|\mathbf{x},\theta,\lambda)p(\mathbf{x}|\theta,\lambda)}{p(s=-1|\theta,\lambda)}\frac{p(s=1|\theta,\lambda)}{p(s=1|\mathbf{x},\theta,\lambda)p(\mathbf{x}|\theta,\lambda)} \tag{10}$$

$$= \frac{p(s=1|\theta,\lambda)}{p(s=-1|\theta,\lambda)}\frac{p(s=-1|\mathbf{x},\theta,\lambda)}{p(s=1|\mathbf{x},\theta,\lambda)} \tag{11}$$

$$= \frac{p(s=1|\theta,\lambda)}{p(s=-1|\theta,\lambda)}\left(\frac{1}{p(s=1|\mathbf{x},\theta,\lambda)}-1\right). \tag{12}$$

The significance of Equation 12 is that it shows how the optimal example weights, the test-to-training ratio $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$, can be determined without knowledge of either training or test density. The right hand side of Equation 12 can be evaluated based on a model that discriminates training against test examples and outputs how much more likely an instance is to occur in the test data than it is to occur in the training data. Instead of potentially high-dimensional densities $p(\mathbf{x}|\theta)$ and $p(\mathbf{x}|\lambda)$, a conditional distribution of the single binary variable s needs to be modeled.

The expression $p(s|\mathbf{x},\theta,\lambda)$ in Equation 12 corresponds to the parametric model $p(s|\mathbf{x};\mathbf{v})$ of Equation 3. With this model we can predict test-to-training density ratios for the training data in $\mathbf{X}_L$ according to Equation 12.

Since our goal is discriminative training, the likelihood function $p(\mathbf{y}|\mathbf{X}_L;\mathbf{w})$ (not taking training-test difference $\mathbf{v}$ into account) would be $\prod_i p(y_i|\mathbf{x}_i;\mathbf{w})$. By using this likelihood $p(\mathbf{y}|\mathbf{X}_L;\mathbf{w})$ instead of $p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})$, one would wrongly assume that the training data $\mathbf{X}_L$ was governed by the test

distribution. Intuitively, $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ dictates how many times, on average, $\mathbf{x}$ should occur in $\mathbf{X}_L$ if $\mathbf{X}_L$ was governed by the test distribution θ. When the individual conditional likelihood of $\mathbf{x}$ is $p(y|\mathbf{x};\mathbf{w})$, then the likelihood of $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ occurrences of $\mathbf{x}$ is $p(y|\mathbf{x};\mathbf{w})^{\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}}$. Using a parametric model $p(s|\mathbf{x};\mathbf{v})$, according to Equation 12 the test-to-training ratio $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$ can be expressed as [1]

$$\frac{p(s=1)}{p(s=-1)}\left(\frac{1}{p(s=1|\mathbf{x};\mathbf{v})}-1\right).$$

Therefore, we define the likelihood function as [2]

$$p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v}) = \prod_{i=1}^{m} p(y_i|\mathbf{x}_i;\mathbf{w})^{\frac{p(s=1)}{p(s=-1)}\left(\frac{1}{p(s_i=1|\mathbf{x}_i;\mathbf{v})}-1\right)}. \tag{13}$$

As an immediate corollary of Manski and Lerman (1977), the likelihood function of Equation 13 has the property that when the true value $\mathbf{v}^*$ is given, its maximizer over $\mathbf{w}$ is a consistent estimator of the true parameter $\mathbf{w}^*$ that has produced labels for the test data under the test distribution θ. That is, as the sample grows, the maximizer of Equation 13 converges in probability to the true value $\mathbf{w}^*$ of parameter $\mathbf{w}$.

For the statistical analysis of case-control studies, Prentice and Pyke (1979) estimate the ratio of two odds ratios with a discriminative model using a formula similar to Equation 12. This double odds ratio is a statistical measure of the relative risk of an incidence (e.g., lung cancer) given a specific exposure (e.g., cigarette smoking) based on data from a retrospective study.

3.3 Optimization Problem for Integrated Model

The likelihood function $p(\mathbf{s}|\mathbf{X};\mathbf{v})$ resolves to $p(s_i=1|\mathbf{x}_i;\mathbf{v})$ for all training instances and $p(s_i=-1|\mathbf{x}_i;\mathbf{v})$ for all test instances:

$$p(\mathbf{s}|\mathbf{X};\mathbf{v}) = \prod_{i=1}^{m} p(s_i=1|\mathbf{x}_i;\mathbf{v}) \prod_{i=m+1}^{m+n} p(s_i=-1|\mathbf{x}_i;\mathbf{v}). \tag{14}$$

Equation 15 summarizes Equations 6 to 8 and Equation 16 inserts the likelihood models (Equations 13 and 14).

$$\begin{aligned} p(\mathbf{w},\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X}) &\propto p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v})p(\mathbf{s}|\mathbf{X};\mathbf{v})p(\mathbf{w})p(\mathbf{v}) && (15)\\ &= \left(\prod_{i=1}^{m} p(y_i|\mathbf{x}_i;\mathbf{w})^{\frac{p(s=1)}{p(s=-1)}\left(\frac{1}{p(s_i=1|\mathbf{x}_i;\mathbf{v})}-1\right)}\right) && (16)\\ &\quad \left(\prod_{i=1}^{m} p(s_i=1|\mathbf{x}_i;\mathbf{v}) \prod_{i=m+1}^{m+n} p(s_i=-1|\mathbf{x}_i;\mathbf{v})\right) p(\mathbf{w})p(\mathbf{v}). \end{aligned}$$

Using a logistic model for $p(s=1|\mathbf{x};\mathbf{v})$, we notice that Equation 12 can be simplified as in Equation 17.

$$\frac{p(s=1)}{p(s=-1)}\left(\frac{1}{1/(1+\exp(-\mathbf{v}^\mathsf{T}\mathbf{x}))}-1\right) = \frac{p(s=1)}{p(s=-1)}\exp(-\mathbf{v}^\mathsf{T}\mathbf{x}). \tag{17}$$

1. For a simplified presentation we drop the conditioning in the prior ratio, that is, $p(s|\theta,\lambda) = p(s)$.

2. The variable s in the prior ratio $\frac{p(s=1)}{p(s=-1)}$ does not need an index i because at this point it is not conditioned on $\mathbf{x}_i$.

Optimization Problem 1 is derived from Equation 16 in logarithmic form, using linear models $\mathbf{v}^\mathsf{T}\mathbf{x}_i$ and $\mathbf{w}^\mathsf{T}\mathbf{x}_i$ and a logistic model for $p(s=1|\mathbf{x};\mathbf{v})$. Negative log-likelihoods are abbreviated $\ell_\mathbf{w}(y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i) = -\log p(y_i|\mathbf{x}_i;\mathbf{w})$ and $\ell_\mathbf{v}(s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i) = -\log p(s_i|\mathbf{x}_i;\mathbf{v})$, respectively; this notation emphasizes the duality between likelihoods and empirical loss functions. The regularization terms correspond to Gaussian priors on $\mathbf{v}$ and $\mathbf{w}$ with variances $\sigma_\mathbf{v}^2$ and $\sigma_\mathbf{w}^2$.

Optimization Problem 1 *Over all* $\mathbf{w}$ *and* $\mathbf{v}$, *minimize*

$$\sum_{i=1}^{m} \frac{p(s=1)}{p(s=-1)} \exp(-\mathbf{v}^\mathsf{T}\mathbf{x}_i)\ell_\mathbf{w}(y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i) + \sum_{i=1}^{m+n} \ell_\mathbf{v}(s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i) + \frac{1}{2\sigma_\mathbf{w}^2}\mathbf{w}^\mathsf{T}\mathbf{w} + \frac{1}{2\sigma_\mathbf{v}^2}\mathbf{v}^\mathsf{T}\mathbf{v}.$$

4. Primal Learning Algorithm

We derive a Newton gradient descent method that directly minimizes Optimization Problem 1 in the attribute space. To this end, we need to derive the gradient and the Hessian of the objective function. The update rule assumes the form of a set of linear equations that have to be solved for the update vector $[\Delta_\mathbf{v}, \Delta_\mathbf{w}]^\mathsf{T}$. It depends on the current parameters $[\mathbf{v}, \mathbf{w}]^\mathsf{T}$, all combinations of training and test data, and resulting coefficients. In order to express the update rule as a single equation in matrix form, we define

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_L & \mathbf{X}_T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{X}_L \end{bmatrix},$$

where $\mathbf{X}_L$ and $\mathbf{X}_T$ are the matrices of training vectors and test vectors, respectively.

Theorem 3 *The update step for the Newton gradient descent minimization of Optimization Problem 1 is* $[\mathbf{v}', \mathbf{w}']^\mathsf{T} \leftarrow [\mathbf{v}, \mathbf{w}]^\mathsf{T} + [\Delta_\mathbf{v}, \Delta_\mathbf{w}]^\mathsf{T}$ *with*

$$(\mathbf{X}\mathbf{\Lambda}\mathbf{X}^\mathsf{T} + \mathbf{S}) \begin{bmatrix} \Delta_\mathbf{v} \\ \Delta_\mathbf{w} \end{bmatrix} = -\mathbf{X}\mathbf{g} - \mathbf{S} \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix}. \tag{18}$$

The definitions of coefficients $\mathbf{\Lambda}$, $\mathbf{S}$, and $\mathbf{g}$—and the proof of the theorem—can be found in Appendix A.

Given the parameter $\mathbf{w}$, a test instance $\mathbf{x}$ is classified as $f(\mathbf{x};\mathbf{w}) = \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x})$.

5. Kernelized Learning Algorithm

We derive a kernelized version of the integrated classifier for differing training and test distributions. A transformation Φ maps instances into a target space in which a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ calculates the inner product $\Phi(\mathbf{x}_i)^\mathsf{T}\Phi(\mathbf{x}_j)$. The update rule (Equation 18) thus becomes

$$(\Phi(\mathbf{X})\mathbf{\Lambda}\Phi(\mathbf{X})^\mathsf{T} + \mathbf{S}) \begin{bmatrix} \Delta_\mathbf{v} \\ \Delta_\mathbf{w} \end{bmatrix} = -\Phi(\mathbf{X})\mathbf{g} - \mathbf{S} \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix}. \tag{19}$$

$\Phi(\mathbf{X})$ is defined by

$$\Phi(\mathbf{X}) = \begin{bmatrix} \Phi(\mathbf{X}_L) & \Phi(\mathbf{X}_T) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Phi(\mathbf{X}_L) \end{bmatrix}.$$

According to the Representer Theorem, the optimal separator is a linear combination of examples. Parameter vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in the dual space weight the influence of all examples:

$$\begin{bmatrix}\mathbf{v}\\ \mathbf{w}\end{bmatrix} = \Phi(\mathbf{X})\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix}.$$

Equation 19 can therefore be rewritten as Equation 20. We now multiply $\Phi(\mathbf{X})^\mathsf{T}$ from the left to both sides and obtain Equation 21. We replace all resulting occurrences of $\Phi(\mathbf{X})^\mathsf{T}\Phi(\mathbf{X})$ by the kernel matrix $\mathbf{K}$ and arrive at Equation 22; $\mathbf{S}$ is replaced by $\mathbf{S}'$ such that $\Phi(\mathbf{X})^\mathsf{T}\mathbf{S}\Phi(\mathbf{X}) = \Phi(\mathbf{X})^\mathsf{T}\Phi(\mathbf{X})\mathbf{S}'$, that is, $S'_{i,i} = \sigma_\mathbf{v}^{-2}$ for $i = 1,\ldots,m+n$ and $S'_{m+n+i,m+n+i} = \sigma_\mathbf{w}^{-2}$ for $i = 1,\ldots,m$. Equation 22 is satisfied when Equation 23 is satisfied. Equation 23 is the update rule for the dual Newton gradient descent.

$$(\Phi(\mathbf{X})\boldsymbol{\Lambda}\Phi(\mathbf{X})^\mathsf{T}+\mathbf{S})\Phi(\mathbf{X})\begin{bmatrix}\Delta_{\boldsymbol{\alpha}}\\ \Delta_{\boldsymbol{\beta}}\end{bmatrix} = -\Phi(\mathbf{X})\mathbf{g} - \mathbf{S}\Phi(\mathbf{X})\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix}, \tag{20}$$

$$\Phi(\mathbf{X})^\mathsf{T}(\Phi(\mathbf{X})\boldsymbol{\Lambda}\Phi(\mathbf{X})^\mathsf{T}+\mathbf{S})\Phi(\mathbf{X})\begin{bmatrix}\Delta_{\boldsymbol{\alpha}}\\ \Delta_{\boldsymbol{\beta}}\end{bmatrix} = -\Phi(\mathbf{X})^\mathsf{T}\Phi(\mathbf{X})\mathbf{g} - \Phi(\mathbf{X})^\mathsf{T}\mathbf{S}\Phi(\mathbf{X})\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix}, \tag{21}$$

$$(\mathbf{K}\boldsymbol{\Lambda}\mathbf{K}+\mathbf{K}\mathbf{S}')\begin{bmatrix}\Delta_{\boldsymbol{\alpha}}\\ \Delta_{\boldsymbol{\beta}}\end{bmatrix} = -\mathbf{K}\mathbf{g} - \mathbf{K}\mathbf{S}'\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix}, \tag{22}$$

$$(\boldsymbol{\Lambda}\mathbf{K}+\mathbf{S}')\begin{bmatrix}\Delta_{\boldsymbol{\alpha}}\\ \Delta_{\boldsymbol{\beta}}\end{bmatrix} = -\mathbf{g} - \mathbf{S}'\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix}. \tag{23}$$

Given the parameters, test instance $\mathbf{x}$ is classified by $f(\mathbf{x};\boldsymbol{\beta}) = \operatorname{sign}(\sum_{i=1}^m \beta_i k(\mathbf{x},\mathbf{x}_i))$.

6. Convexity Analysis and Solving the Optimization Problems

The following theorem specifies sufficient conditions for convexity of Optimization Problem 1. With this theorem we can easily check whether the integrated classifier for covariate shift is convex for specific models of the negative log-likelihood functions. The negative log-likelihood function $\ell_\mathbf{w}$ itself and its first and second derivatives are needed. Equations 31 and 32 in Appendix A define shorthand notation which we will use in the following.

Theorem 4 *Optimization Problem 1 is convex if the loss function $\ell_\mathbf{v}$ is convex and $\ell_\mathbf{w}$ is log-convex and non-negative. The log-convexity condition is equivalent to*

$$\ell_\mathbf{w}\ell''_\mathbf{w} - \ell'^2_\mathbf{w} \geq 0. \tag{24}$$

Proof Looking at Optimization Criterion 1 we immediately see that the regularizers are convex. If $\ell_\mathbf{v}$ is convex, the second term is convex as well. We therefore only need to analyze the convexity of the term

$$\sum_{i=1}^m \frac{p(s=1)}{p(s=-1)}\exp(-\mathbf{v}^\mathsf{T}\mathbf{x}_i)\ell_\mathbf{w}(y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i).$$

A sum is convex if the single summands are convex. And a sufficient condition for convexity of a function is that it is non-negative and log-convex. The above expression is non-negative as $\ell_\mathbf{w}$ is non-negative. This means we only need to check whether

$$\log\frac{p(s=1)}{p(s=-1)} - \mathbf{v}^\mathsf{T}\mathbf{x}_i + \log\ell_{\mathbf{w},i}$$

is convex. The prior ratio is assumed to be constant. The second term is linear and therefore convex and the third term is the log-convexity condition of $\ell_{\mathbf{w}}$. The second derivative of $\log \ell_{\mathbf{w}}$ is

$$\ell_{\mathbf{w}}^{-1}\ell_{\mathbf{w}}'' + \ell_{\mathbf{w}}^{-2}\ell_{\mathbf{w}}'^2,$$

thus $\log \ell_{\mathbf{w}}$ is convex if $\ell_{\mathbf{w}}\ell_{\mathbf{w}}'' - \ell_{\mathbf{w}}'^2$ is non-negative. ■

In order to check Optimization Criterion 1 for convexity we need to choose models of the negative log-likelihood $\ell_{\mathbf{v}}$ and $\ell_{\mathbf{w}}$ and derive their first and second derivatives. These derivations are also needed to actually minimize Optimization Criterion 1 with the Newton update steps derived in the last section.

We use a logistic model $\ell_{\mathbf{v}}(s_i\mathbf{v}^\mathsf{T}\mathbf{x}) = \log(1+\exp(-s_i\mathbf{v}^\mathsf{T}\mathbf{x}))$; the abbreviations of Appendix A can now be expanded:

$$\ell'_{\mathbf{v},i}s_ix_{ij} = -\frac{\exp(-s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i)}{1+\exp(-s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i)}s_ix_{ij}; \quad \ell''_{\mathbf{v},i}x_{ij}x_{ik} = \frac{\exp(-s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i)}{(1+\exp(-s_i\mathbf{v}^\mathsf{T}\mathbf{x}_i))^2}x_{ij}x_{ik}.$$

For the target classifier, we detail the derivations for logistic and for exponential models of $\ell_{\mathbf{w}}$. For the logistic model the derivatives of $\ell_{\mathbf{w}}$ are the same as for $\ell_{\mathbf{v}}$, only $\mathbf{v}$ needs to be replaced by $\mathbf{w}$ and s_i by y_i. For an exponential model with $\ell_{\mathbf{w}}(y_i\mathbf{w}^\mathsf{T}\mathbf{x}) = \exp(-y_i\mathbf{w}^\mathsf{T}\mathbf{x})$ the abbreviations are expanded as follows:

$$\ell'_{\mathbf{w},i}y_ix_{ij} = -\exp(-y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i)y_ix_{ij}; \quad \ell''_{\mathbf{w},i}x_{ij}x_{ik} = \exp(-y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i)x_{ij}x_{ik}.$$

Using Theorem 4 we can now easily check the convexity of the integrated classifier with logistic model and with exponential model for $\ell_{\mathbf{w}}$.

Corollary 5 *With a logistic model for $\ell_{\mathbf{w}}$, the condition of Equation 24 is violated and therefore Optimization Problem 1 with logistic model for $\ell_{\mathbf{w}}$ may not convex in general.*

Proof Inserting the logistic function into Equation 24 we get the following solution.

$$\ell_{\mathbf{w},i}\ell''_{\mathbf{w},i} - \ell'^2_{\mathbf{w},i} = \frac{\exp(-y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i)}{(1+\exp(-y_i\mathbf{w}^\mathsf{T}\mathbf{x}))^2}\left(\log(1+\exp(-y_i\mathbf{w}^\mathsf{T}\mathbf{x})) - \exp(-y_i\mathbf{w}^\mathsf{T}\mathbf{x}_i)\right). \tag{25}$$

The fraction in Equation 25 is always positive, the difference term is always negative which violates the condition of Equation 24. ■

Empirically, we find that it is a good choice to select the parameters of a regular, *iid* logistic regression classifier as starting point for the Newton gradient search.

One can easily show that the condition of Equation 24 is violated when $\ell_{\mathbf{w}}$ is chosen as hinge loss or quadratic loss.

Corollary 6 *Optimization Problem 1 with exponential model for $\ell_{\mathbf{w}}$ is convex.*

Proof The exponential loss is non-negative and its logarithm is linear and therefore convex. ■

This means the global optimum of Optimization Problem 1 with exponential model for $\ell_{\mathbf{w}}$ can easily be found by Newton gradient descent.

7. Two-Stage Approximation to Integrated Model

The previous sections describe a complete solution to the learning problem under covariate shift. Optimization Problem 1 is convex for the exponential model; solving it using the efficient procedures derived in Sections 4 and 5 produces a globally optimal solution.

For the logistic model, unfortunately, convexity cannot be guaranteed. Furthermore, the regularized regression classifier is deeply embedded in Optimization Problem 1. It would not be easy to replace it by a different type of classifier such as, for instance, a decision tree. We will now discuss an approximation to Optimization Problem 1 which solves two consecutive optimization problems. The first optimization problem produces example-specific weights; the second step generates a classifier from the weighted examples. Both optimization problems are convex for exponential, logistic, and hinge loss as well as for many other loss functions. But most significantly, the two-stage approximation is conceptually simple: the second optimization step can be carried out by any learning procedure that is able to scale the loss incurred by each example using prescribed weight factors. Example-specific weights can easily be incorporated into virtually any learning method. Furthermore, as a result of the decomposition into two optimization problems parameter tuning becomes much easier because cross-validation can be used (cf. Section 9).

The derivation in Section 3.1 approximates the integral over $\mathbf{v}$ by simultaneously selecting a pair of values which maximize the posterior. This leads to the joint MAP hypothesis over $\mathbf{v}$ and $\mathbf{w}$. In the resulting optimization problem, $\mathbf{v}$ and $\mathbf{w}$ are free parameters. At a higher degree of approximation, one may factorize the posterior (Equation 26) and at first approximate the integral over $\mathbf{v}$ by the maximum of $p(\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X})$ (Equations 29 and 30). Subsequently, the posterior over $\mathbf{w}$ is maximized given fixed parameters $\mathbf{v}_{\text{MAP}'}$ (Equations 27 and 28).

$$\begin{aligned}
\mathbf{w}^* &= \operatorname{argmax}_{\mathbf{w}} \int p(\mathbf{w},\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X}) d\mathbf{v} \\
&= \operatorname{argmax}_{\mathbf{w}} \int p(\mathbf{w}|\mathbf{y},\mathbf{s},\mathbf{X};\mathbf{v}) p(\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X}) d\mathbf{v} && (26) \\
&\approx \operatorname{argmax}_{\mathbf{w}} p(\mathbf{w}|\mathbf{y},\mathbf{s},\mathbf{X};\mathbf{v}_{\text{MAP}'}) && (27) \\
&= \operatorname{argmax}_{\mathbf{w}} p(\mathbf{y}|\mathbf{s},\mathbf{X};\mathbf{w},\mathbf{v}_{\text{MAP}'}) p(\mathbf{w}) && (28) \\
\text{with} \quad \mathbf{v}_{\text{MAP}'} &= \operatorname{argmax}_{\mathbf{v}} p(\mathbf{v}|\mathbf{y},\mathbf{s},\mathbf{X}) && (29) \\
&= \operatorname{argmax}_{\mathbf{v}} p(\mathbf{s}|\mathbf{y},\mathbf{X};\mathbf{v}) p(\mathbf{v}). && (30)
\end{aligned}$$

This results in two optimization problems. Only parameter $\mathbf{v}$ is free in the first stage (Optimization Problem 2). The test-to-training ratio (Equation 17) can be derived from the resulting value of $\mathbf{v}$.

Optimization Problem 2 *Over* $\mathbf{v}$*, minimize*

$$\sum_{i=1}^{m+n} \ell_{\mathbf{v}}(s_i \mathbf{v}^\mathsf{T} \mathbf{x}_i) + \frac{1}{2\sigma_{\mathbf{v}}^2} \mathbf{v}^\mathsf{T} \mathbf{v}.$$

In the second stage (Optimization Problem 3), the target model parameters $\mathbf{w}$ are optimized with constant parameters $\mathbf{v}$ and constant example weights. The parameters $\mathbf{v}$ are the result of Optimization Problem 2.

Optimization Problem 3 *Over* $\mathbf{w}$ *(*$\mathbf{v}$ *is constant), minimize*

$$\sum_{i=1}^{m} \frac{p(s=1)}{p(s=-1)} \exp(-\mathbf{v}^\mathsf{T} \mathbf{x}_i) \ell_{\mathbf{w}}(y_i \mathbf{w}^\mathsf{T} \mathbf{x}_i) + \frac{1}{2\sigma_{\mathbf{w}}^2} \mathbf{w}^\mathsf{T} \mathbf{w}.$$

The criterion of Optimization Problem 3 weights the loss $\ell_{\mathbf{w}}(y_i \mathbf{w}^\mathsf{T} \mathbf{x}_i)$ that each example incurs such that the sample is matched to the test distribution. The last term $\frac{1}{2\sigma_{\mathbf{w}}^2} \mathbf{w}^\mathsf{T} \mathbf{w}$ is the regularizer of the regression. Optimization Problem 3 can easily be adapted to virtually any type of classification mechanism by inserting the appropriate loss function $\ell_{\mathbf{w}}(y_i \mathbf{w}^\mathsf{T} \mathbf{x}_i)$ and regularizer. Operationally, an arbitrary classification procedure is applied to a sample that is either resampled from the training data according to sampling distribution $\frac{p(s=1)}{p(s=-1)} \exp(-\mathbf{v}^\mathsf{T} \mathbf{x}_i)$, or the classifier is applied to the training data with the example-specific loss scaled according to $\frac{p(s=1)}{p(s=-1)} \exp(-\mathbf{v}^\mathsf{T} \mathbf{x}_i)$.

8. Relationship to Kernel Mean Matching

Huang et al. (2007) motivate the kernel mean matching algorithm as a procedure that minimizes the distance between the means of unlabeled and weighted labeled data in feature space. If the kernel is universal this is equivalent to minimizing the difference of the distributions. We derive a new interpretation for kernel mean matching that shows its relation to Optimization Problem 2 and the above two-stage approximation.

Using a hinge loss for $\ell_{\mathbf{v}}(s_i(\mathbf{v}^\mathsf{T} \mathbf{x}_i + b))$ in Optimization Problem 2 and an explicit offset parameter b we obtain a regular support vector machine. The kernel matrix of this SVM is $\begin{bmatrix} \mathbf{K}_{(LL)} & \mathbf{K}_{(LT)} \\ \mathbf{K}_{(LT)}^\mathsf{T} & \mathbf{K}_{(TT)} \end{bmatrix}$ and the target variables are $s_i \in \{-1, 1\}$. An SVM can heuristically be simplified by setting the dual parameters α_i for the unlabeled examples to a fixed value $\frac{m}{n}$. This can be interpreted as a mixture between an SVM and a Rocchio classifier. The α_i corresponding to the labeled examples ($s_i = 1$) are trained with an SVM; setting α_i of all unlabeled examples ($s_i = -1$) to $\frac{m}{n}$ approximates the negative class (the unlabeled examples) by their centroid in feature space in accordance with the Rocchio classifier (Joachims, 1997).

The SVM optimization criterion with fixed $\alpha_i = \frac{m}{n}$ for examples with $s_i = -1$ is

$$\begin{aligned} \min_{\alpha_L} \quad & \tfrac{1}{2} \alpha_L^\mathsf{T} \mathbf{K}_{(LL)} \alpha_L - \tfrac{m}{n} \alpha_L^\mathsf{T} \mathbf{K}_{(LT)} \mathbf{1} + \tfrac{1}{2} \tfrac{m^2}{n^2} \mathbf{1} \mathbf{K}_{(TT)} \mathbf{1} - \alpha_L^\mathsf{T} \mathbf{1} - n \tfrac{m}{n} \\ & \text{subject to } \alpha_i \in \left[0, \sigma_{\mathbf{v}}^2\right] \text{ and } \textstyle\sum_{i=1}^m \alpha_i = \sum_{i=1}^n \frac{m}{n} = m; \end{aligned}$$

vector α_L denotes all elements α_i with $i = 1, \ldots, m$. We can drop the constant terms ($\alpha_L^\mathsf{T} \mathbf{1}$ is constant because of the second constraint) and arrive at Optimization Problem 4.

Optimization Problem 4

$$\min_{\alpha_L} \quad \tfrac{1}{2} \alpha_L^\mathsf{T} \mathbf{K}_{(LL)} \alpha_L - \tfrac{m}{n} \alpha_L^\mathsf{T} \mathbf{K}_{(LT)} \mathbf{1} \quad \text{subject to } \alpha_i \in \left[0, \sigma_{\mathbf{v}}^2\right] \text{ and } \alpha_L^\mathsf{T} \mathbf{1} = m.$$

This is the dual objective of kernel mean matching. The only difference is that Huang et al. (2007) relax the second constraint up to a small constant ε, their constraint is $m(1-\varepsilon) \leq \alpha_L^\mathsf{T} \mathbf{1} \leq m(1+\varepsilon)$. Empirically we find that setting ε to zero has no impact on the performance. The parameter $\sigma_{\mathbf{v}}^2$ corresponds to parameter B in Equation 2.

In order to solve the second stage (Optimization Problem 3), kernel mean matching does not use re-weighting factors $\frac{p(s=1)}{p(s=-1)} \exp(-\mathbf{v}^\mathsf{T} \mathbf{x}_i - b)$ but directly uses the dual α_i parameters as weights.

To sum up, kernel mean matching can be interpreted as a variant of Optimization Problem 2. It discriminates training against test examples using a partially Rocchio-style approximation to the SVM optimization criterion.

9. Parameter Tuning

Optimization Problem 1 relies on hyper-parameters $\sigma_{\mathbf{v}}^2$ and $\sigma_{\mathbf{w}}^2$ that need to be tuned. For the two-stage approximation of Section 7 and reference methods like kernel mean matching two similar parameters need to be specified. In addition to the regularization parameters kernel parameters need to be tuned for non-linear kernels. Parameter tuning for covariate shift models is much more difficult than for regular prediction models because in the covariate shift setting there is no labeled data available drawn from the test distribution. Parameter tuning by regular cross-validation on the labeled training data is inappropriate because the labeled training data is not governed by the test distribution.

In the following paragraphs we describe different tuning procedures; two procedures require prior knowledge and one does not require prior knowledge on the hyper-parameters. The tuning procedures with prior knowledge can be used for all described models. The one without prior knowledge cannot be used for kernel mean matching and the one-stage model of Optimization Problem 1.

A typical setting with prior knowledge on the hyper-parameters is when the difference between training and test data is introduced by a covariate shift over time and the input distribution shifts constantly over time. The most recent data is the unlabeled test data and the older data has been labeled and is the training data. In this setting the parameters can be tuned by splitting the labeled training data into two consecutive parts. The tuning models are learned on the part with earlier timestamps and the hyper-parameters $\sigma_{\mathbf{v}}^2$ and $\sigma_{\mathbf{w}}^2$ and kernel parameters are optimized on the part with later timestamps.

Another setting with prior knowledge is when in addition to the pair of training and test set an additional pair of training and fully labeled test set from a different domain with a similar magnitude of covariate shift is available. This additional set can be used to tune the parameters. Due to the similar magnitude of the covariate shift the optimal parameters for the additional domain are assumed to be a good choice for the parameters of the target domain.

For some two-stage models for covariate shift there is no prior knowledge necessary to tune hyper-parameters. Sugiyama et al. (2008) propose to tune the regularizer of the KLIEP model with cross-validation. In this manner the first stage parameter $\sigma_{\mathbf{v}}^2$ (and kernel parameters) of the two-stage model of Section 7 can be tuned as follows. The training and the test data are both split into training and tuning folds and the hold-out likelihood of the tuning folds is optimized with grid search on $\sigma_{\mathbf{v}}^2$ (and kernel parameters). The hold-out likelihood measures the predictive performance of the model $p(s|\mathbf{x};\mathbf{v})$ with respect to predicting the selector variable s of the hold out examples. Once the regularizer of the first stage is tuned, the second stage parameter $\sigma_{\mathbf{w}}^2$ (and kernel parameters) can be tuned with cross-validation on weighted training data (Sugiyama and Müller, 2005). The data of training folds as well as the data of tuning folds are weighted with the estimated training-to-test ratio.

Kernel mean matching does not provide out-of-sample predictions and it is therefore difficult to tune the regularization parameter B with cross-validation. The one-stage model of Optimization Problem 1 is also difficult to tune with cross-validation because there is a bidirectional influence between the parameters $\sigma_{\mathbf{v}}^2$ and $\sigma_{\mathbf{w}}^2$.

In order to compare the one-stage model and kernel mean matching to the other two-stage models we use tuning procedures based on prior knowledge in the empirical studies in the next section.

10. Empirical Results

We study the benefit of two versions of the integrated classifier for covariate shift and other reference methods on spam filtering, text classification, and landmine detection problems. The first integrated classifier uses a logistic model for $\ell_{\mathbf{w}}$ ("integrated log model"), the second an exponential model for $\ell_{\mathbf{w}}$ ("integrated exp model"); $\ell_{\mathbf{v}}$ is a logistic model in both cases.

The first baseline is a classifier trained under *iid* assumption with logistic $\ell_{\mathbf{w}}$. All other reference methods consist of a two-stage procedure: first, the difference between training and test distribution is estimated, the classifier is trained on weighted data in a second step. The second method is kernel mean matching (Huang et al., 2007); we set $\varepsilon = \sqrt{m-1}/\sqrt{m}$ as proposed by the authors. In the third method, separate density estimates for $p(\mathbf{x}|\lambda)$ and $p(\mathbf{x}|\theta)$ are obtained using kernel density estimation (Shimodaira, 2000), the bandwidth of the kernel is chosen according to the rule-of-thumb of Silverman (1986).

The last two reference methods rely on the two-stage approximation of Optimization Problems 2 and 3 with a logistic regression ("two-stage LR") and an exponential model classifier ("two-stage exp model") as their second stages. The example weights are computed according to Equation 17 using a logistic model in the first stage, $p(s=1|\mathbf{x};\mathbf{v})$ is estimated by training a logistic regression that discriminates training from test examples.

The baselines differ in the first stage, the second stage is based on a logistic regression classifier with weighted examples in all cases but the two-stage exponential model baseline. We use a maximum likelihood estimate of $\frac{m}{n}$ for $\frac{p(s=1)}{p(s=-1)}$. We use tuning procedures that rely on prior knowledge (cf. Section 9). Short descriptions of the respective tuning data can be found below. For all experiments we tune the regularization parameters of all methods (and the variance parameter of the RBF kernels for the landmine experiments) by maximizing AUC on the tuning set.

We use the spam filtering data of Bickel et al. (2007); the collection contains nine different inboxes with test emails (5270 to 10964 emails, depending on inbox) and one set of training emails compiled from various different sources. We use a fixed set of 1000 emails as training data. We randomly select between 32 and 2048 emails from one of the original inboxes. We repeat this process 10 times for 2048 test emails and 20 to 640 times for 1024 to 32 test emails. As tuning data we use the labeled emails from an additional inbox different from the test inboxes.

The performance measure is the rate by which the $1-\text{AUC}$ risk is reduced over the *iid* baseline (Bickel and Scheffer, 2007); it is computed as $1-\frac{1-\text{AUC}}{1-\text{AUC}_{iid}}$. We use linear kernels for all methods. We analyze the rank of the kernel matrix and find that it fulfills the universal kernel requirement of kernel mean matching; this is due to the high-dimensionality of the data.

Figure 1 (top row) shows the results for various numbers of unlabeled examples. The left column of Figure 1 compares the integrated classifiers for covariate shift to the kernel mean matching and kernel density estimation baselines. The right column compares the integrated classifiers (Optimization Problem 1) with the two-stage approximations (Optimization Problems 2 and 3). The results for a specific number of unlabeled examples are averaged over 10 to 640 random test samples and averaged over all nine inboxes. Averaged over all users and inbox sizes the absolute AUC of the *iid* classifier is 0.994. Error bars indicate standard errors of the $1-\text{AUC}$ risk.

The integrated and two-step logistic regression and exponential models and kernel mean matching perform similarly well. The differences to the *iid* baseline are highly significant. For 1048 examples the $1-\text{AUC}$ risk is even reduced by an average of 30% with the integrated exponential model classifier! The kernel density estimation procedure is not able to beat the *iid* baseline. The

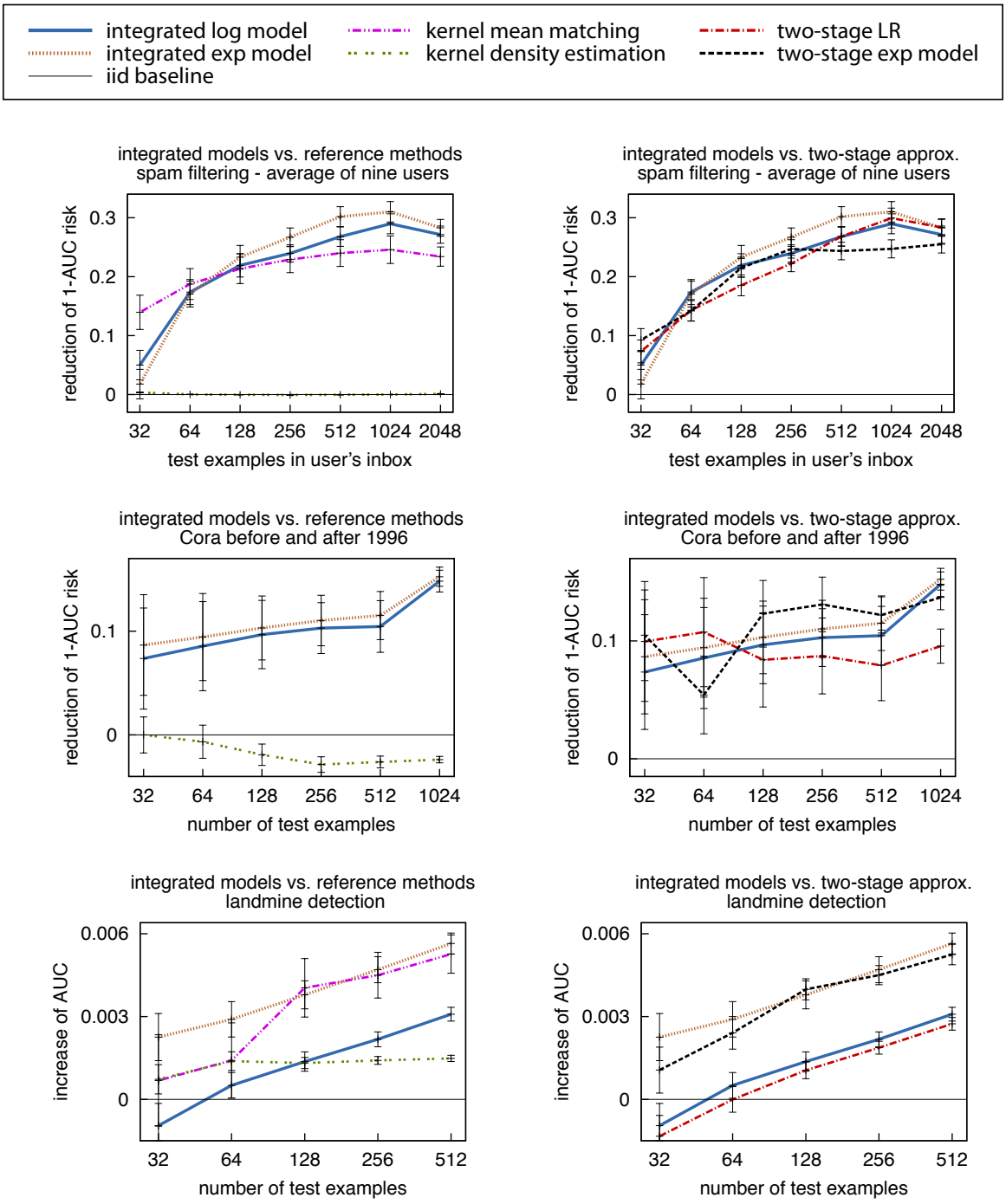


Figure 1: Average reduction of $1 - \text{AUC}$ risk over nine users for spam filtering (top row) and Cora *Machine Learning/Networking* classification before and after 1996 (second row) and average increase of AUC for landmine detection over 812 pairs of mine fields (bottom row) depending on the number of unlabeled test examples.

convex integrated exponential model performs slightly better than its two-stage approximation; for larger number of test examples (512 to 2048) this difference is statistically significant according to a paired t-test with significance level of 5%. For the logistic model, the two-stage optimization performs similarly well as the integrated version.

We now study text classification using computer science papers from the Cora data set. The task is to discriminate Machine Learning from Networking papers. We select 812 papers written before 1996 from both classes as training examples and 1285 papers written after 1996 as test examples. For parameter tuning we apply an additional time split on the training data; we train on the papers written before 1995 and tune on papers written 1995 (cf. Section 9).

Title and abstract are transformed into *tfidf* vectors, the number of distinct words is about 40,000. We again use linear kernels (rank analysis verifies the universal kernel property) and average the results over 20 to 640 random test samples for different sizes (1024 for 20 samples to 32 for 640 samples) of test sets. The resulting $1-\text{AUC}$ risk is shown in Figure 1 (second row). The average absolute AUC of the *iid* classifier is 0.998. The methods based on discriminative density estimates significantly outperform all other methods. Kernel mean matching is not displayed because its average performance lies far below the *iid* baseline. The integrated models reduce the $1-\text{AUC}$ risk by 15% for 1024 test examples.

In a third set of experiments we study the problem of detecting landmines using the data set of Xue et al. (2007). The collection contains data of 29 mine fields in different regions. Binary labels (landmine or safe ground) and nine dimensional feature vectors extracted from radar images are provided. There are about 500 examples for each mine field. Each of the fields has a distinct distribution of input patterns, varying from highly foliated to desert areas.

We enumerate all 29×28 pairs of mine fields, using one field as training and the other as test data. For tuning we hold out 4 of the 812 pairs. Results are increases over the *iid* baseline, averaged over all $29 \times 28 - 4$ combinations. We use RBF kernels with kernel width 0.3 for all methods. The results are displayed in Figure 1 (bottom row). The average absolute AUC of the *iid* baseline is 0.64 with a standard deviation of 0.07; note, that the error bars are much smaller than the absolute standard deviation because they indicate the standard error of the *differences* to the *iid* baseline.

For this problem, the exponential model classifiers and kernel mean matching significantly outperform all other methods on average. Considering only methods with logistic target model, kernel mean matching is better than all other methods. Integrated logistic regression and two-stage logistic regression are still significantly better than the *iid* baseline except for 32 and 64 test examples. The integrated classifiers are slightly better than the two-stage variants.

11. Conclusion

We derived a discriminative model for learning under differing training and test distributions. The contribution of each training instance to the optimization problem ideally needs to be weighted with its test-to-training density ratio. We show that this ratio can be expressed—without modeling either training or test density—by a discriminative model that characterizes how much more likely an instance is to occur in the test sample than it is to occur in the training sample.

We described a generative model whose parameters can be estimated with a joint MAP hypothesis of both the parameters of the test-to-training model and the final classifier. Optimizing these dependent parameters sequentially incurs an additional approximation compared to solving the joint optimization problem. We derived a primal and a kernelized Newton gradient descent procedure for

the joint optimization problem. Theorem 4 specifies the condition for the convexity of Optimization Problem 1. Checking the condition using popular loss functions as models of the negative log-likelihoods reveals that Optimization Problem 1 is convex with exponential loss.

We gave a new interpretation for kernel mean matching and show that it is also based on a discriminative model similar to Optimization Problem 2.

Empirically, we found that the integrated and the two-stage models as well as kernel mean matching outperform the *iid* baseline and the kernel density estimation model in almost all cases. In some cases, the integrated models perform slightly better than their two-stage counterparts. The performance of kernel mean matching depends on the problem; for one out of three problems it did not beat the *iid* baseline, for the others it yielded comparable results to the integrated models.

The two-stage model is conceptually simpler than the integrated model, and may in some cases have the greatest practical utility. The main advantage compared to the integrated model is that regularization parameters can be tuned without prior knowledge by cross-validation. Another advantage of the two-stage model is that in the second stage, after the example-specific weights have been derived, virtually any learning mechanism can be employed to produce the final classifier from the weighted training sample. This comes at the cost of only a marginal loss of performance compared to the integrated model.

Acknowledgments

We gratefully acknowledge support from the German Science Foundation DFG and from STRATO AG. We thank Google for supporting us with a Google Research Award. We also thank the anonymous reviewers for their helpful comments. We wish to thank Jiayuan Huang, Alex Smola, Arthur Gretton, Karsten Borgward, and Bernhard Schölkopf who provided their implementation of the kernel mean matching algorithm.

Appendix A. Newton Gradient Descent—Proof of Theorem 3

In this Appendix, we derive Newton gradient descent updates for Optimization Problem 1 and thereby prove Theorem 3. We abbreviate

$$\ell_{\mathbf{v},i} = \ell_{\mathbf{v}}(s_i \mathbf{v}^\mathsf{T} \mathbf{x}_i); \;\; \ell'_{\mathbf{v},i} s_i x_{ij} = \frac{\partial \ell_{\mathbf{v}}(s_i \mathbf{v}^\mathsf{T} \mathbf{x}_i)}{\partial v_j}; \;\; \ell''_{\mathbf{v},i} x_{ij} x_{ik} = \frac{\partial^2 \ell_{\mathbf{v}}(s_i \mathbf{v}^\mathsf{T} \mathbf{x}_i)}{\partial v_j v_k}; \tag{31}$$

$$\ell_{\mathbf{w},i} = \ell_{\mathbf{w}}(y_i \mathbf{w}^\mathsf{T} \mathbf{x}_i); \;\; \ell'_{\mathbf{w},i} y_i x_{ij} = \frac{\partial \ell_{\mathbf{w}}(y_i \mathbf{w}^\mathsf{T} \mathbf{x}_i)}{\partial w_j}; \;\; \ell''_{\mathbf{w},i} x_{ij} x_{ik} = \frac{\partial^2 \ell_{\mathbf{w}}(y_i \mathbf{w}^\mathsf{T} \mathbf{x}_i)}{\partial w_j w_k}; \tag{32}$$

$$\omega_i = \frac{p(s=1)}{p(s=-1)} \exp(-\mathbf{v}^\mathsf{T} \mathbf{x}_i)$$

and denote the objective function of Optimization Problem 1 by

$$F(\mathbf{v}, \mathbf{w}) \;\; = \;\; \sum_{i=1}^{m} \omega_i \ell_{\mathbf{w},i} + \sum_{i=1}^{m+n} \ell_{\mathbf{v},i} + \frac{1}{2\sigma_{\mathbf{w}}^2} \mathbf{w}^\mathsf{T} \mathbf{w} + \frac{1}{2\sigma_{\mathbf{v}}^2} \mathbf{v}^\mathsf{T} \mathbf{v}.$$

We compute the gradient with respect to $\mathbf{v}$ and $\mathbf{w}$.

$$\begin{aligned}\frac{\partial F(\mathbf{v},\mathbf{w})}{\partial v_j} &= -\sum_{i=1}^{m}\omega_i\ell_{\mathbf{w},i}x_{ij}+\sum_{i=1}^{m+n}\ell'_{\mathbf{v},i}s_ix_{ij}+\frac{1}{\sigma_{\mathbf{v}}^2}v_j\\ \frac{\partial F(\mathbf{v},\mathbf{w})}{\partial w_j} &= \sum_{i=1}^{m}\omega_i\ell'_{\mathbf{w},i}y_ix_{ij}+\frac{1}{\sigma_{\mathbf{w}}^2}w_j.\end{aligned}$$

The Hessian is the matrix of second derivatives.

$$\begin{aligned}\frac{\partial^2 F(\mathbf{v},\mathbf{w})}{\partial v_j\partial v_k} &= \sum_{i=1}^{m}\omega_i\ell_{\mathbf{w},i}x_{ij}x_{ik}+\sum_{i=1}^{m+n}\ell''_{\mathbf{v},i}x_{ij}x_{ik}+\frac{1}{\sigma_{\mathbf{v}}^2}\delta_{jk}\\ \frac{\partial^2 F(\mathbf{v},\mathbf{w})}{\partial v_j\partial w_k} &= -\sum_{i=1}^{m}\omega_i\ell'_{\mathbf{w},i}y_ix_{ij}x_{ik}\\ \frac{\partial^2 F(\mathbf{v},\mathbf{w})}{\partial w_j\partial w_k} &= \sum_{i=1}^{m}\omega_i\ell''_{\mathbf{w},i}x_{ij}x_{ik}+\frac{1}{\sigma_{\mathbf{w}}^2}\delta_{jk}.\end{aligned}$$

We can rewrite the gradient as $\mathbf{X}\mathbf{g}+\mathbf{S}\begin{bmatrix}\mathbf{v}\\ \mathbf{w}\end{bmatrix}$ and the Hessian as $\mathbf{X}\mathbf{\Lambda}\mathbf{X}^\mathsf{T}+\mathbf{S}$ using the following definitions, where d is the dimensionality of $\mathbf{X}_T$ and $\mathbf{X}_L$.

$$\begin{aligned}g_i &= -\omega_i\ell_{\mathbf{w},i}+\ell'_{\mathbf{v},i} && \text{for} \quad i=1,\ldots,m;\\ g_{m+i} &= -\ell'_{\mathbf{v},m+i} && \text{for} \quad i=1,\ldots,n;\\ g_{m+n+i} &= \omega_i\ell'_{\mathbf{w},i}y_i && \text{for} \quad i=1,\ldots,m;\\ S_{i,i} &= \sigma_{\mathbf{v}}^{-2} && \text{for} \quad i=1,\ldots,d;\\ S_{d+i,d+i} &= \sigma_{\mathbf{w}}^{-2} && \text{for} \quad i=1,\ldots,d;\end{aligned}$$

$$\mathbf{\Lambda}=\begin{bmatrix}\operatorname*{diag}_{i=1,\ldots,m}\left(\omega_i\ell_{\mathbf{w},i}+\ell''_{\mathbf{v},i}\right) & \mathbf{0} & -\operatorname*{diag}_{i=1,\ldots,m}\left(\omega_i\ell'_{\mathbf{w},i}y_i\right)\\ \mathbf{0} & \operatorname*{diag}_{i=1,\ldots,n}\left(\ell''_{\mathbf{v},m+i}\right) & \mathbf{0}\\ -\operatorname*{diag}_{i=1,\ldots,m}\left(\omega_i\ell'_{\mathbf{w},i}y_i\right) & \mathbf{0} & \operatorname*{diag}_{i=1,\ldots,m}\left(\omega_i\ell''_{\mathbf{w},i}\right)\end{bmatrix}.$$

The update step for the Newton gradient descent minimization of Optimization Problem 1 is $[\mathbf{v}',\mathbf{w}']^\mathsf{T}\leftarrow[\mathbf{v},\mathbf{w}]^\mathsf{T}+[\Delta_\mathbf{v},\Delta_\mathbf{w}]^\mathsf{T}$ with

$$(\mathbf{X}\mathbf{\Lambda}\mathbf{X}^\mathsf{T}+\mathbf{S})\begin{bmatrix}\Delta_\mathbf{v}\\ \Delta_\mathbf{w}\end{bmatrix}=-\mathbf{X}\mathbf{g}-\mathbf{S}\begin{bmatrix}\mathbf{v}\\ \mathbf{w}\end{bmatrix}.$$

References

S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems*, 2007.

S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the International Conference on Machine Learning*, 2007.

C. Cortes, M. Mohri, M. Riley, and A. Rostamizadeh. Sample selection bias correction theory. In *Proceedings of the International Conference on Algorithmic Learning Theory*, 2008.

M. Dudik, R. Schapire, and S. Phillips. Correcting sample selection bias in maximum entropy density estimation. In *Advances in Neural Information Processing Systems*, 2005.

C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the International Joint Conference on Artificial Intellligence*, 2001.

J. Heckman. Sample selection bias as a specification error. *Econometrica*, 47:153–161, 1979.

J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*, 2007.

N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6:429–449, 2002.

T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, 1997.

J. Lunceford and M. Davidian. Stratification and weighting via the propensity score in estimation of causal treatment effects: a comparative study. *Statistics in Medicine*, 23(19):2937–2960, 2004.

C. Manski and S. Lerman. The estimation of choice probabilities from choice based samples. *Econometrica*, 45(8):1977–1988, 1977.

R. Prentice and R. Pyke. Logistic disease incidence models and case-control studies. *Biometrika*, 66(3):403–411, 1979.

P. Rosenbaum and D. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244, 2000.

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

M. Sugiyama and K.-R. Müller. Input-dependent estimation of generalization error under covariate shift. *Statistics and Decision*, 23(4):249–279, 2005.

M. Sugiyama, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*, 2008.

J. Tsuboi, H. Kashima, S. Hido, S. Bickel, and M. Sugiyama. Direct density ratio estimation for large-scale covariate shift adaptation. In *Proceedings of the SIAM International Conference on Data Mining*, 2008.

Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the International Conference on Machine Learning*, 2004.

Journal of Machine Learning Research 10 (2009) 2157-2192 Submitted 2/09; Published 10/09

Optimized Cutting Plane Algorithm for Large-Scale Risk Minimization*

Vojtěch Franc XFRANCV@CMP.FELK.CVUT.CZ
Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Technicka 2, 166 27 Praha 6,
Czech Republic

Sören Sonnenburg SOEREN.SONNENBURG@TUEBINGEN.MPG.DE
Friedrich Miescher Laboratory
Max Planck Society
Spemannstr. 39
72076 Tübingen, Germany

Editor: Michele Sebag

Abstract

We have developed an optimized cutting plane algorithm (OCA) for solving large-scale risk minimization problems. We prove that the number of iterations OCA requires to converge to a ε precise solution is approximately linear in the sample size. We also derive OCAS, an OCA-based linear binary Support Vector Machine (SVM) solver, and OCAM, a linear multi-class SVM solver. In an extensive empirical evaluation we show that OCAS outperforms current state-of-the-art SVM solvers like $\text{SVM}^{\text{light}}$, SVM^{perf} and BMRM, achieving speedup factor more than 1,200 over $\text{SVM}^{\text{light}}$ on some data sets and speedup factor of 29 over SVM^{perf} , while obtaining the same precise support vector solution. OCAS, even in the early optimization steps, often shows faster convergence than the currently prevailing approximative methods in this domain, SGD and Pegasos. In addition, our proposed linear multi-class SVM solver, OCAM, achieves speedups of factor of up to 10 compared to $\text{SVM}^{\text{multi-class}}$. Finally, we use OCAS and OCAM in two real-world applications, the problem of human acceptor splice site detection and malware detection. Effectively parallelizing OCAS, we achieve state-of-the-art results on an acceptor splice site recognition problem only by being able to learn from all the available 50 million examples in a 12-million-dimensional feature space. Source code, data sets and scripts to reproduce the experiments are available at `http://cmp.felk.cvut.cz/~xfrancv/ocas/html/`.

Keywords: risk minimization, linear support vector machine, multi-class classification, binary classification, large-scale learning, parallelization

1. Introduction

Many applications in, for example, bioinformatics, IT-security and text classification come with *huge* numbers (e.g., millions) of data points, which are indeed *necessary* to obtain state-of-the-

*. A large part of the work was done while VF and SS were at the Fraunhofer Institute FIRST, Kekulestr. 7, 12489 Berlin, Germany.

art results. They, therefore, require extremely efficient computational methods capable of dealing with ever growing data sizes. A wide range of machine learning methods can be described as the unconstrained regularized risk minimization problem

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}\in\Re^n} F(\mathbf{w}) := \frac{1}{2}\|\mathbf{w}\|^2 + CR(\mathbf{w}), \tag{1}$$

where $\mathbf{w} \in \Re^n$ denotes the parameter vector to be learned, $\frac{1}{2}\|\mathbf{w}\|^2$ is a quadratic regularization term, $C > 0$ is a fixed regularization constant and $R\colon \Re^n \to \Re$ is a non-negative convex risk function approximating the empirical risk (e.g., training error).

Special cases of problem (1) are, for example, support vector classification and regression (e.g., Cortes and Vapnik, 1995; Cristianini and Shawe-Taylor, 2000), logistic regression (Collins et al., 2000), maximal margin structured output classification (Tsochantaridis et al., 2005), SVM for multivariate performance measures (Joachims, 2005), novelty detection (Schölkopf et al., 1999), learning Gaussian processes (Williams, 1998) and many others.

Problem (1) has usually been solved in its dual formulation, which typically only uses the computation of dot products between examples. This enables the use of kernels that implicitly compute the dot product between examples in a Reproducing Kernel Hilbert Space (RKHS) (Schölkopf and Smola, 2002). On the one hand, solving the dual formulation is efficient when dealing with high-dimensional data. On the other hand, efficient and accurate solvers for optimizing the kernelized dual formulation for large sample sizes do not exist.

Recently, focus has shifted towards methods optimizing problem (1) directly in the primal. Using the primal formulation is efficient when the number of examples is very large and the dimensionality of the input data is moderate or the inputs are sparse. This is typical in applications like text document analysis and bioinformatics, where the inputs are strings embedded into a sparse high-dimensional feature space, for example, by using the bag-of-words representation. A way to exploit the primal formulation for learning in the RKHS is based on decomposing the kernel matrix and thus effectively linearizing the problem (Schölkopf and Smola, 2002).

Due to its importance, the literature contains a plethora of specialized solvers dedicated to particular risk functions $R(\mathbf{w})$ of problem (1). Binary SVM solvers employing the hinge risk $R(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i\langle \mathbf{w}, \mathbf{x}_i\rangle\}$ especially have been studied extensively (e.g., Joachims, 1999; Zanni et al., 2006; Chang and Lin, 2001; Sindhwani and Keerthi, 2007; Chapelle, 2007; Lin et al., 2007). Recently, Teo et al. (2007) proposed the Bundle Method for Risk Minimization (BMRM), which is a general approach for solving problem (1). BMRM is not only a general but also a highly modular solver that only requires two specialized procedures, one to evaluate the risk $R(\mathbf{w})$ and one to compute its subgradient. BMRM is based on iterative approximation of the risk term by cutting planes. It solves a reduced problem obtained by substituting the cutting plane approximation of the risk into the original problem (1). Teo et al. (2007) compared BMRM with specialized solvers for SVM classification, SV regression and ranking, and reported promising results. However, we will show that the implementation of the cutting plane algorithm (CPA) used in BMRM can be drastically sped up making it efficient even for large-scale SVM binary and multi-class classification problems.

In this paper, we develop an *efficient and general* algorithm to solve the regularized risk minimization problem (1). Building on the work of Teo et al. (2007), we propose an *optimized cutting plane algorithm* (OCA) that extends the standard CPA in two ways. First, unlike CPA, we use the solution of the reduced problem as a direction in the line-search to directly minimize the original

(master) problem (1). Second, we introduce a new cutting plane selection strategy that reduces the number of cutting planes required to achieve the prescribed precision and thus significantly speeds up convergence. An *efficient* line-search procedure for the optimization of (1) is the *only additional requirement* of OCA compared to the standard CPA (or BMRM).

While our proposed method (OCA) is applicable to a wide range of risk terms R, we will—due to their importance—discuss in more detail two special cases: learning of the binary (two-class) and multi-class SVM classifiers. We show that the line-search procedure required by OCA can be solved exactly for both the binary and multi-class SVM problems in $O(m \log m)$ and $O(m \cdot Y^2 + m \cdot Y \log(m \cdot Y))$ time, respectively, where m is the number of examples and Y is the number of classes. We abbreviate OCA for binary SVM classifiers with OCAS and the multi-class version with OCAM.

We perform an extensive experimental evaluation of the proposed methods, OCAS and OCAM, on several problems comparing them with the current state of the art. In particular, we would like to highlight the following experiments and results:

- We compare OCAS with the state-of-the-art solvers for binary SVM classification on previously published data sets. We show that OCAS significantly outperforms the competing approaches achieving speedups factors of more than 1,200.
- We evaluate OCAS using the large-scale learning challenge data sets and evaluation protocols described in Sonnenburg et al. (2009). Although OCAS is an implementation of a general method for risk minimization (1), it is shown to be competitive with dedicated binary SVM solvers, which ultimately won the large-scale learning challenge.
- We demonstrate that OCAS can be sped up by efficiently parallelizing its core subproblems.
- We compare OCAM with the state-of-the-art implementation of the CPA-based solver for training multi-class SVM classifiers. We show that OCAM achieves speedups of an order of magnitude.
- We show that OCAS and OCAM achieve state-of-the-art recognition performance for two real-world applications. In the first application, we attack a splice site detection problem from bioinformatics. In the second, we address the problem of learning a malware behavioral classifier used in computer security systems.

The OCAS solver for training the binary SVM classification was published in our previous paper (Franc and Sonnenburg, 2008a). This paper extends the previous work in several ways. First, we formulate OCA for optimization of the general risk minimization problem (1). Second, we give an improved convergence proof for the general OCA (in Franc and Sonnenburg 2008a the upper bound on the number of iterations as a function of precision ε scales with $O(\frac{1}{\varepsilon^2})$, while in this paper the bound is improved to $O(\frac{1}{\varepsilon})$). Third, we derive a new instance of OCA for training the multi-class SVM classifiers. Fourth, the experiments are extended by (i) including the comparison on the challenge data sets and using the challenge protocol, (ii) performing experiments on multi-class classification problems and (iii) solving two real-world applications from bioinformatics and computer security.

The remainder of this paper is organized as follows. The standard cutting plane algorithm (CPA) to solve (1) is reviewed in Section 2. In Section 3, we point out a source of inefficiency

of CPA and propose a new optimized cutting plane algorithm (OCA) to drastically reduce training times. We then develop OCA for two special cases linear binary SVMs (OCAS, see Section 3.1) and linear multiclass SVMs (OCAM, see Section 3.2). In Section 4, we empirically show that using OCA, training times can be drastically reduced on a wide range of large-scale data sets including the challenge data sets. Finally, we attack two real-world applications. First, in Section 5.1, we apply OCAS to a human acceptor splice site recognition problem achieving state-of-the art results by training on all available sequences—a data set of 50 million examples (itself about 7GB in size) using a 12 million dimensional feature space. Second, in Section 5.2, we apply OCAM to learn a behavioral malware classifier and achieve a speedup of factor of 20 compared to the previous approach and a speedup of factor of 10 compared to the state-of-the-art implementation of the CPA. Section 6 concludes the paper.

2. Cutting Plane Algorithm

In CPA terminology, the original problem (1) is called the master problem. Using the approach of Teo et al. (2007), one may define a reduced problem of (1) which reads

$$\mathbf{w}_t = \operatorname*{argmin}_{\mathbf{w}} F_t(\mathbf{w}) := \left[\frac{1}{2}\|\mathbf{w}\|^2 + CR_t(\mathbf{w})\right]. \tag{2}$$

(2) is obtained from master problem (1) by substituting a piece-wise linear approximation R_t for the risk R. Note that only the risk term R is approximated while the regularization term $\frac{1}{2}\|\mathbf{w}\|^2$ remains unchanged. The idea is that in practise one usually needs only a small number of linear terms in the piece-wise linear approximation R_t to adequately approximate the risk R around the optimum $\mathbf{w}^*$. Moreover, it was shown in Teo et al. (2007) that the number of linear terms needed to achieve arbitrary precise approximation does not depend on the number of examples.

We now derive the approximation to R. Since the risk term R is a convex function, it can be approximated at any point $\mathbf{w}'$ by a linear under estimator

$$R(\mathbf{w}) \geq R(\mathbf{w}') + \langle \mathbf{a}', \mathbf{w} - \mathbf{w}' \rangle, \qquad \forall \mathbf{w} \in \Re^n, \tag{3}$$

where $\mathbf{a}' \in \partial R(\mathbf{w}')$ is any subgradient of R at the point $\mathbf{w}'$. We will use a shortcut $b' = R(\mathbf{w}') - \langle \mathbf{a}', \mathbf{w}' \rangle$ to abbreviate (3) as $R(\mathbf{w}) \geq \langle \mathbf{a}', \mathbf{w} \rangle + b'$. In CPA terminology, $\langle \mathbf{a}', \mathbf{w} \rangle + b' = 0$ is called a cutting plane.

To approximate the risk R better than by using a single cutting plane, we can compute a collection of cutting planes $\{\langle \mathbf{a}_i, \mathbf{w} \rangle + b_i = 0 \mid i = 1, \ldots, t\}$ at t distinct points $\{\mathbf{w}_1, \ldots, \mathbf{w}_t\}$ and take their point-wise maximum

$$R_t(\mathbf{w}) = \max\left\{0, \max_{i=1,\ldots,t} \left(\langle \mathbf{a}_i, \mathbf{w} \rangle + b_i\right)\right\}. \tag{4}$$

The zero cutting plane is added to the maximization as the risk R is assumed to be non-negative. The subscript $_t$ denotes the number of cutting planes used in the approximation R_t. It follows directly from (3) that the approximation R_t is exact at the points $\{\mathbf{w}_1, \ldots, \mathbf{w}_t\}$ and that R_t lower bounds R, that is, that $R(\mathbf{w}) \geq R_t(\mathbf{w}), \forall \mathbf{w} \in \Re^n$ holds. In turn, the objective function F_t of the reduced problem lower bounds the objective F of the master problem.

Having readily computed R_t, we may now use it in the reduced problem (2). Substituting (4) with (2), the reduced problem can be expressed as a linearly constrained quadratic problem

$$(\mathbf{w}_t, \xi_t) := \operatorname*{argmin}_{\mathbf{w} \in \Re^n, \xi \in \Re} \left[\frac{1}{2}\|\mathbf{w}\|^2 + C\xi\right], \tag{5}$$

subject to

$$\xi \geq 0\,, \quad \xi \geq \langle \mathbf{a}_i, \mathbf{w} \rangle + b_i\,, \quad i = 1, \ldots, t\,.$$

The number of constraints in (5) equals the number of cutting planes t which can be drastically lower than the number of constraints in the master problem (1) when expressed as a constrained QP task. As the number of cutting planes is typically much smaller than the data dimensionality n, it is convenient to solve the reduced problem (5) by optimizing its dual formulation, which reads

$$\alpha_t := \operatorname*{argmax}_{\alpha \in \mathcal{A}_t} D_t(\alpha) := \Big[\sum_{i=1}^{t} \alpha_i b_i - \frac{1}{2} \Big\| \sum_{i=1}^{t} \mathbf{a}_i \alpha_i \Big\|^2 \Big]\,, \tag{6}$$

where $\mathcal{A}_t$ is a convex feasible set containing all vectors $\alpha \in \Re^t$ satisfying

$$\sum_{i=1}^{t} \alpha_i \leq C\,, \qquad \alpha_i \geq 0\,, i = 1, \ldots, t\,.$$

The dual formulation contains only t variables bound by $t+1$ constraints of simple form. Thus task (6) can be efficiently optimized by standard QP solvers. Having (6) solved, the primal solution can be computed as

$$\mathbf{w}_t = -\sum_{i=1}^{t} \mathbf{a}_i [\alpha_t]_i\,, \quad \text{and} \quad \xi_t = \max_{i=1,\ldots,t} \left(\langle \mathbf{w}_t, \mathbf{a}_i \rangle + b_i \right).$$

Solving the reduced problem is beneficial if we can effectively select a small number of cutting planes such that the solution of the reduced problem is sufficiently close to the master problem. CPA selects the cutting planes using a simple strategy described by Algorithm 1.

Algorithm 1 Cutting Plane Algorithm (CPA)

1: $t := 0$.
2: **repeat**
3: Compute $\mathbf{w}_t$ by solving the reduced problem (5).
4: Add a new cutting plane to approximate the risk R at the current solution $\mathbf{w}_t$, that is, compute $\mathbf{a}_{t+1} \in \partial R(\mathbf{w}_t)$ and $b_{t+1} := R(\mathbf{w}_t) - \langle \mathbf{a}_{t+1}, \mathbf{w}_t \rangle$.
5: $t := t + 1$
6: **until** a stopping condition is satisfied.

The algorithm is very general. To use it for a particular problem one only needs to supply a formula to compute the cutting plane as required in Step 4, that is, formulas for computing the subgradient $\mathbf{a} \in \partial R(\mathbf{w})$ and the objective value $R(\mathbf{w})$ at given point $\mathbf{w}$.

It is natural to stop the algorithm when

$$F(\mathbf{w}_t) - F_t(\mathbf{w}_t) \leq \varepsilon \tag{7}$$

holds. Because $F_t(\mathbf{w}_t)$ is the lower bound of the optimal value, $F(\mathbf{w}^*)$, it follows that a solution $\mathbf{w}_t$ satisfying (7) also guarantees $F(\mathbf{w}_t) - F(\mathbf{w}^*) \leq \varepsilon$, that is, the objective value differs from the optimal one by ε at most. An alternative stopping condition advocated in Joachims (2006) stops the algorithm when $R(\mathbf{w}_t) - R_t(\mathbf{w}_t) \leq \hat{\varepsilon}$. It can be seen that the two stopping conditions become equivalent if we set $\varepsilon = C\hat{\varepsilon}$. Hence we will consider only the former stopping condition (7).

Theorem 1 by Teo et al. (2007) guarantees convergence of the CPA algorithm in $O(\frac{1}{\varepsilon})$ time for a broad class of risk functions:

Theorem 1 *(Teo et al., 2007) Assume that $\|\partial R(\mathbf{w})\| \leq G$ for all $\mathbf{w} \in \mathcal{W}$, where $\mathcal{W}$ is some domain of interest containing all $\mathbf{w}_{t'}$ for $t' \leq t$. In this case, for any $\varepsilon > 0$ and $C > 0$, Algorithm 1 satisfies the stopping condition (7) after at most*

$$\log_2 \frac{F(\mathbf{0})}{4C^2G^2} + \frac{8C^2G^2}{\varepsilon} - 2$$

iterations.

3. Optimized Cutting Plane Algorithm (OCA)

We first point out a source of inefficiency in CPA and then propose a new method to alleviate the problem.

CPA selects a new cutting plane such that the reduced problem objective function $F_t(\mathbf{w}_t)$ monotonically increases w.r.t. the number of iterations t. However, there is no such guarantee for the master problem objective $F(\mathbf{w}_t)$. Even though it will ultimately converge arbitrarily close to the minimum $F(\mathbf{w}^*)$, its value can heavily fluctuate between iterations (Figure 1). The reason for these

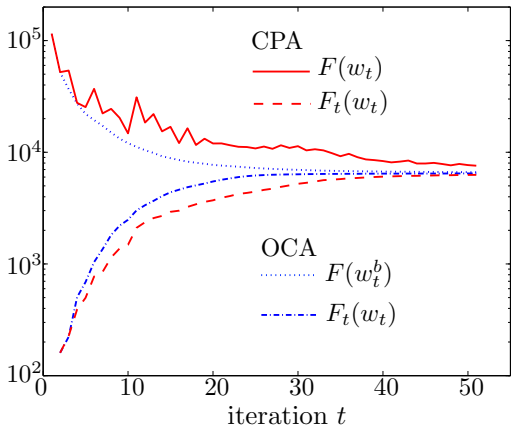


Figure 1: Convergence behavior of the standard CPA vs. the proposed OCA.

fluctuations is that at each iteration t, CPA selects the cutting plane that perfectly approximates the master objective F *at the current solution* $\mathbf{w}_t$. However, there is no guarantee that such a cutting plane will be an active constraint in the vicinity of the optimum $\mathbf{w}^*$, nor must the new solution $\mathbf{w}_{t+1}$ of the reduced problem improve the master objective. In fact, it often occurs that $F(\mathbf{w}_{t+1}) > F(\mathbf{w}_t)$. As a result, a lot of the selected cutting planes do not contribute to the approximation of the master objective around the optimum which, in turn, increases the number of iterations.

To speed up the convergence of CPA, we propose a new method which we call the *optimized cutting plane algorithm* (OCA). Unlike standard CPA, OCA aims at simultaneously optimizing the master and reduced problems' F and F_t objective functions, respectively. In addition, OCA tries to select cutting planes that have a higher chance of actively contributing to the approximation of the master objective function F around the optimum $\mathbf{w}^*$. In particular, we propose the following three changes to CPA.

Change 1 We maintain the best-so-far best solution $\mathbf{w}_t^{\mathrm{b}}$ obtained during the first t iterations, that is, $F(\mathbf{w}_1^{\mathrm{b}}),\ldots,F(\mathbf{w}_t^{\mathrm{b}})$ forms a monotonically decreasing sequence.

Change 2 The new best-so-far solution $\mathbf{w}_t^{\mathrm{b}}$ is found by searching along a line starting at the previous solution $\mathbf{w}_{t-1}^{\mathrm{b}}$ and crossing the reduced problem's solution $\mathbf{w}_t$, that is,

$$\mathbf{w}_t^{\mathrm{b}} = \min_{k\geq 0} F(\mathbf{w}_{t-1}^{\mathrm{b}}(1-k)+\mathbf{w}_t k)\,. \tag{8}$$

Change 3 The new cutting plane is computed to approximate the master objective F at a point $\mathbf{w}_t^{\mathrm{c}}$ which lies in a vicinity of the best-so-far solution $\mathbf{w}_t^{\mathrm{b}}$. In particular, the point $\mathbf{w}_t^c$ is computed as

$$\mathbf{w}_t^c = \mathbf{w}_t^{\mathrm{b}}(1-\mu)+\mathbf{w}_t\mu\,, \tag{9}$$

where $\mu\in(0,1]$ is a prescribed parameter. Having the point $\mathbf{w}_t^c$, the new cutting plane is given by $\mathbf{a}_{t+1}\in\partial R(\mathbf{w}_t^c)$ and $b_{t+1}=R(\mathbf{w}_t^c)-\langle\mathbf{a}_{t+1},\mathbf{w}_t^c\rangle$.

Algorithm 2 describes the proposed OCA. Figure 1 shows the impact of the proposed changes to the convergence. OCA generates a monotonically decreasing sequence of master objective values and a monotonically and strictly increasing sequence of reduced objective values, that is,

$$F(\mathbf{w}_1^{\mathrm{b}})\geq\ldots\geq F(\mathbf{w}_t^{\mathrm{b}})\,,\quad\text{and}\quad F_1(\mathbf{w}_1)<\ldots<F_t(\mathbf{w}_t)\,.$$

Note that for CPA only the latter is satisfied. Similar to CPA, a natural stopping condition for OCA reads

$$F(\mathbf{w}_t^{\mathrm{b}})-F_t(\mathbf{w}_t)\leq\varepsilon\,, \tag{10}$$

where $\varepsilon>0$ is a prescribed precision parameter. Satisfying the condition (10) guarantees that $F(\mathbf{w}_t^{\mathrm{b}})-F(\mathbf{w}^*)\leq\varepsilon$ holds.

Algorithm 2 Optimized Cutting Plane Algorithm (OCA)

1: Set $t:=0$ and $\mathbf{w}_0^{\mathrm{b}}:=\mathbf{0}$.
2: **repeat**
3: Compute $\mathbf{w}_t$ by solving the reduced problem (5).
4: Compute a new best-so-far solution $\mathbf{w}_t^{\mathrm{b}}$ using the line-search (8).
5: Add a new cutting plane: compute $\mathbf{a}_{t+1}\in\partial R(\mathbf{w}_t^c)$ and $b_{t+1}:=R(\mathbf{w}_t^c)-\langle\mathbf{a}_{t+1},\mathbf{w}_t^c\rangle$ where $\mathbf{w}_t^{\mathrm{c}}$ is given by (9).
6: $t:=t+1$
7: **until** a stopping condition is satisfied

Theorem 2 *Assume that $\|\partial R(\mathbf{w})\|\leq G$ for all $\mathbf{w}\in\mathcal{W}$, where $\mathcal{W}$ is some domain of interest containing all $\mathbf{w}_{t'}$ for $t'\leq t$. In this case, for any $\varepsilon>0$, $C>0$ and $\mu\in(0,1]$, Algorithm 2 satisfies the stopping condition (10) after at most*

$$\log_2\frac{F(\mathbf{0})}{4C^2G^2}+\frac{8C^2G^2}{\varepsilon}-2$$

iterations.

Theorem 2 is proven in Appendix A. Finally, there are two relevant remarks regarding Theorem 2:

Remark 1 Although Theorem 2 holds for any μ from the interval $(0,1]$ its particular value has impact on the convergence speed in practice. We found experimentally (see Section 4.1) that $\mu = 0.1$ works consistently well throughout experiments.

Remark 2 Note that the bound on the maximal number of iterations of OCA given in Theorem 2 coincides with the bound for CPA in Theorem 1. Despite the same theoretical bounds, in practice OCA converges significantly faster compared to CPA, achieving speedups of more than one order of magnitude as will be demonstrated in the experiments (Section 4). In the convergence analysis (see Appendix A) we give an intuitive explanation of why OCA converges faster than CPA.

In the following subsections we will use the OCA Algorithm 2 to develop efficient binary linear and multi-class SVM solvers. To this end, we develop fast methods to solve the problem-dependent subtasks, the line-search step (step 4 in Algorithm 2) and the addition of a new cutting plane (step 5 in Algorithm 2).

3.1 Training Linear Binary SVM Classifiers

Given an example set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \in (\Re^n \times \{-1,+1\})^m$, the goal is to find a parameter vector $\mathbf{w} \in \Re^n$ of the liner classification rule $h(\mathbf{x}) = \mathrm{sgn}\langle \mathbf{w}, \mathbf{x} \rangle$. The parameter vector $\mathbf{w}$ is obtained by minimizing

$$F(\mathbf{w}) := \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\}\,, \tag{11}$$

which is a special instance of the regularized risk minimization problem (1) with the risk

$$R(\mathbf{w}) := \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\}\,. \tag{12}$$

It can be seen that (12) is a convex piece-wise linear approximation of the training error $\frac{1}{m}\sum_{i=1}^{m} [\![h(\mathbf{x}_i) \neq y_i]\!]$.

To use the OCA Algorithm 2 for solving (12), we need the problem-dependent steps 4 and 5. First, we need to supply a procedure performing the line-search (8) as required in Step 4. Section 3.1.1 describes an efficient algorithm solving the line-search exactly in $O(m \log m)$ time. Second, Step 5 requires a formula for computing a subgradient $\mathbf{a} \in \partial R(\mathbf{w})$ of the risk (12) which reads

$$\mathbf{a} = -\frac{1}{m}\sum_{i=1}^{m} \pi_i y_i \mathbf{x}_i\,, \qquad \pi_i = \begin{cases} 1 & \text{if} \quad y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \leq 1\,, \\ 0 & \text{if} \quad y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 1\,. \end{cases}$$

Both the line-search and computation of the subgradient can be sped up via the parallelization described in Section 3.1.2. We call the resulting algorithm the *optimized cutting plane algorithm for SVM* (OCAS).

3.1.1 LINE-SEARCH FOR LINEAR BINARY SVM CLASSIFIERS

The line-search (8) requires minimization of a univariate convex function

$$F(\mathbf{w}_{t-1}^{\mathrm{b}}(1-k)+\mathbf{w}_t k) = \frac{1}{2}\|\mathbf{w}_{t-1}^{\mathrm{b}}(1-k)+\mathbf{w}_t k\|^2 + CR(\mathbf{w}_{t-1}^{\mathrm{b}}(1-k)+\mathbf{w}_t k)\,, \tag{13}$$

with R defined by (12). Note that the line-search very much resembles the master problem (1) with one-dimensional data. We show that the line-search can be solved exactly in $O(m\log m)$ time.

We abbreviate $F(\mathbf{w}_{t-1}^{\mathrm{b}}(1-k)+\mathbf{w}_t k)$ by $f(k)$ which is defined as

$$f(k) := f_0(k) + \sum_{i=1}^{m} f_i(k) = \frac{1}{2}k^2 A_0 + kB_0 + C_0 + \sum_{i=1}^{m} \max\{0, kB_i + C_i\}\,,$$

where

$$\begin{array}{rclcrcl} A_0 &=& \|\mathbf{w}_{t-1}^{\mathrm{b}} - \mathbf{w}_t\|^2 & & & & \\ B_0 &=& \langle \mathbf{w}_{t-1}^{\mathrm{b}}, \mathbf{w}_t - \mathbf{w}_{t-1}^{\mathrm{b}}\rangle\,, & B_i &=& \frac{C}{m} y_i \langle \mathbf{x}_i, \mathbf{w}_{t-1}^{\mathrm{b}} - \mathbf{w}_t\rangle\,, & i=1,\ldots,m\,, \\ C_0 &=& \frac{1}{2}\|\mathbf{w}_{t-1}^{\mathrm{b}}\|^2\,, & C_i &=& \frac{C}{m}(1 - y_i\langle \mathbf{x}_i, \mathbf{w}_{t-1}^{\mathrm{b}}\rangle)\,, & i=1,\ldots,m\,. \end{array} \tag{14}$$

Hence the line-search (8) involves solving $k^* = \operatorname{argmin}_{k\geq 0} f(k)$ and computing $\mathbf{w}_t^{\mathrm{b}} = \mathbf{w}_{t-1}^{\mathrm{b}}(1-k^*) + \mathbf{w}_t k^*$. Since $f(k)$ is a convex function, its unconstrained minimum is attained at the point k^*, at which the sub-differential $\partial f(k)$ contains zero, that is, $0 \in \partial f(k^*)$ holds. The subdifferential of f reads

$$\partial f(k) = kA_0 + B_0 + \sum_{i=1}^{m} \partial f_i(k)\,, \quad \text{where} \quad \partial f_i(k) = \begin{cases} 0 & \text{if} \quad kB_i + C_i < 0\,, \\ B_i & \text{if} \quad kB_i + C_i > 0\,, \\ [0, B_i] & \text{if} \quad kB_i + C_i = 0\,. \end{cases}$$

Note that the subdifferential is not a function because there exist k's for which $\partial f(k)$ is an interval. The first term of the subdifferential $\partial f(k)$ is an ascending linear function $kA_0 + B_0$ since A_0 must be greater than zero. Note that $A_0 = \|\mathbf{w}_{t-1}^{\mathrm{b}} - \mathbf{w}_t\|^2$ equals 0 only if the algorithm has converged to the optimum $\mathbf{w}^*$, but in this case the line-search is not invoked. The term $\partial f_i(k)$ appearing in the sum is

	$k < k_i$	$k = k_i$	$k > k_i$
$B_i = 0$	0	0	0
$B_i < 0$	B_i	$[B_i, 0]$	0
$B_i > 0$	0	$[0, B_i]$	B_i

Table 1: The value of $\partial f_i(k)$ with respect to k.

either constantly zero, if $B_i = 0$, or it is a step-like jump whose value changes at the point $k_i = -\frac{C_i}{B_i}$. In particular, the value of $\partial f_i(k)$ w.r.t. k is summarized in Table 1. Hence the subdifferential $\partial f(k)$ is a monotonically increasing function as is illustrated in Figure 2. To solve $k^* = \operatorname{argmin}_{k\geq 0} f(k)$ we proceed as follows:

1. We compute the maximal value of the subdifferential $\partial f(k)$ at point 0:

$$\max(\partial f(0)) = B_0 + \sum_{i=1}^{m} [\![(B_i < 0 \wedge k_i > 0) \vee (B_i > 0 \wedge k_i \leq 0)]\!] B_i\,.$$

2. If $\max(\partial f(0))$ is strictly greater than zero, we know that the unconstrained minimum $\operatorname{argmin}_k f(k)$ is attained at a point less than or equal to 0. Thus, the constrained minimum $k^* = \operatorname{argmin}_{k\geq 0} f(k)$, that is, the result of the line-search, is attained at the point $k^* = 0$.

3. If $\max(\partial f(0))$ is less than zero, then the optimum $k^* = \operatorname{argmin}_{k\geq 0} f(k)$ corresponds to the unconstrained optimum $\operatorname{argmin}_k f(k)$. To get k^* we need to find an intersection between the graph of $\partial f(k)$ and the x-axis. This can be done efficiently by sorting points $K = \{k_i \mid k_i > 0, i = 1,\ldots,m\}$ and checking the condition $0 \in \partial f(k)$ for $k \in K$ and for k in the intervals which split the domain $(0,\infty)$ in the points K. These computations are dominated by sorting the numbers K, which takes $O(|K|\log|K|)$ time.

Computing the parameters (14) of the function $f(k)$ requires $O(mn)$ time, where m is the number of examples and n is the number of features. Having the parameters computed, the worst-case time complexities of the steps 1, 2 and 3 are $O(m)$, $O(1)$ and $O(m\log m)$, respectively.

3.1.2 PARALLELIZATION

Apart from solving the reduced problem (2), all subtasks of OCAS can be efficiently parallelized:

Output computation. This involves computation of the dot products $\langle \mathbf{w}_t, \mathbf{x}_i \rangle$ for all $i = 1,\ldots,m$, which requires $O(s)$ time, where s equals the number of all non-zero elements in the training examples. Distributing the computation equally to p processors reduces the effort to $O(\frac{s}{p})$. Note that the remaining products with data required by OCAS, that is, $\langle \mathbf{w}_t^{\mathrm{b}}, \mathbf{x}_i \rangle$ and $\langle \mathbf{w}_t^{\mathrm{c}}, \mathbf{x}_i \rangle$, can be computed from $\langle \mathbf{w}_t, \mathbf{x}_i \rangle$ in time $O(m)$.

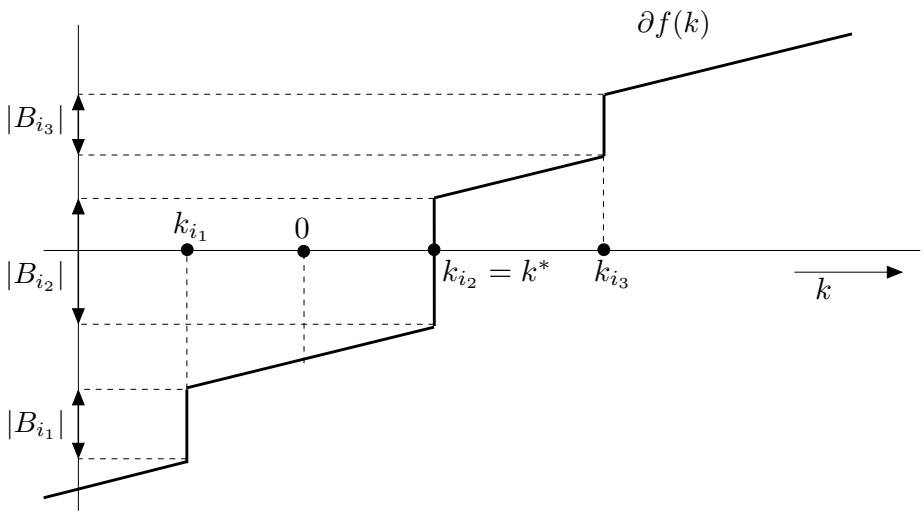


Figure 2: Graph depicting the subdifferential $\partial f(k)$ of the objective function $f(k)$. The line-search requires computing $k^* = \min_{k\geq 0} f(k)$ which is equivalent to finding the intersection k^* between the graph of $\partial f(k)$ and the positive part of the x-axis.

Line-search. The dominant part is sorting $|K|$ numbers which can be done in $O(|K|\log|K|)$ time. A speedup can be achieved by parallelizing the sorting function by using p processors, reducing time complexity to $O\big(\frac{|K|\log|K|}{p}\big)$. Note that our implementation of OCAS uses quicksort, whose worst-case time complexity is $O(|K|^2)$, although its *expected* run-time is $O(|K|\log|K|)$.

Cutting plane computation. The dominant part requires the sum $-\frac{1}{m}\sum_{i=1}^{m}\pi_i y_i \mathbf{x}_i$, which can be done in $O(s_\pi)$ time, where $s_\pi = |\{i|\pi_i \neq 0,\ \forall i = 1,\ldots,m\}|$ is the number of non-zero π_i. Using p processors results in a time complexity of $O(\frac{s_\pi}{p})$.

It is worth mentioning that OCAS usually requires a small number of iterations (usually less than 100 and almost always less than 1000). Hence, solving the reduced problem, which cannot be parallelized, is not the bottleneck, especially when the number of examples m is large.

3.2 Training General Linear Multi-Class SVM Classifiers

So far we have assumed that (i) the ultimate goal is to minimize the probability of misclassification, (ii) the input observations $\mathbf{x}$ are vectors from $\Re^n$ and (iii) the label y can attain only two values $\{-1,+1\}$. In this section, we will consider the regularized risk minimization framework applied to the learning of a general linear classifier (Tsochantaridis et al., 2005).

We assume that the input observation x is from an arbitrary set $\mathcal{X}$ and the label y can have values from $\mathcal{Y} = \{1,\ldots,Y\}$. In addition, let $\delta\colon \mathcal{Y}\times\mathcal{Y}\to\Re$ be an arbitrary loss function which satisfies $\delta(y,y) = 0, \forall y\in\mathcal{Y}$, and $\delta(y,y') > 0, \forall (y,y')\in\mathcal{Y}\times\mathcal{Y}, y\neq y'$. We consider the multi-class classification rule $h\colon \mathcal{X}\to\mathcal{Y}$ defined as

$$h(x;\mathbf{w}) = \operatorname*{argmax}_{y\in\mathcal{Y}} \langle \mathbf{w}, \Psi(x,y)\rangle\,,$$

where $\mathbf{w}\in\Re^d$ is a parameter vector and $\Psi\colon \mathcal{X}\times\mathcal{Y}\to\Re^d$ is an arbitrary map from the input-output space to the parameter space. Given example set $\{(x_1,y_1),\ldots,(x_m,y_m)\}\in(\mathcal{X}\times\mathcal{Y})^m$, learning the parameter vector $\mathbf{w}$ using the regularized risk minimization framework requires solving problem (1) with the empirical risk $R_{\text{emp}}(h(\cdot;\mathbf{w})) = \frac{1}{m}\sum_{i=1}^{m}\delta(h(x_i),y_i)$. Tsochantaridis et al. (2005) propose two convex piece-wise linear upper bounds on risk $R_{\text{emp}}(h(\cdot;\mathbf{w}))$. The first one, called the *margin re-scaling* approach, defines the proxy risk as

$$R(\mathbf{w}) := \frac{1}{m}\sum_{i=1}^{m}\max_{y\in\mathcal{Y}}\big(\delta(y,y_i) + \langle\Psi(x_i,y) - \Psi(x_i,y_i),\mathbf{w}\rangle\big)\,. \tag{15}$$

The second one, called the *slack re-scaling* approach, defines the proxy risk as

$$R(\mathbf{w}) := \frac{1}{m}\sum_{i=1}^{m}\max_{y\in\mathcal{Y}}\delta(y,y_i)\big(1 + \langle\Psi(x_i,y) - \Psi(x_i,y_i),\mathbf{w}\rangle\big)\,. \tag{16}$$

In the rest of this section we will derive the OCA solver for minimization of the margin re-scaling risk (15). Note that modification of the solver to optimize the slack re-scaling risk (16) is straightforward and that both variants have exactly the same computational complexity. Note also that for the special case when $\delta(y,y')$ is the 0/1-loss, both (15) and (16) become equivalent.

To use the OCA Algorithm 2 for the regularized minimization of (15), we need, first, to derive a procedure performing the line-search (8) required in Step 4 and, second, to derive a formula for the computation of the subgradient of the risk R as required in Step 5. Section 3.2.1 describes an efficient algorithm solving the line-search exactly in $O(m \cdot Y^2 + m \cdot Y \log(m \cdot Y))$ time. The formula for computing the subgradient $\mathbf{a} \in \partial R(\mathbf{w})$ of the risk (15) reads

$$\mathbf{a} = \frac{1}{m} \sum_{i=1}^{m} \left(\Psi(x_i, \hat{y}_i) - \Psi(x_i, y_i) \right),$$

where

$$\hat{y}_i = \operatorname*{argmax}_{y \in \mathcal{Y}} \left(\delta(y_i, y) + \langle \Psi(x_i, y) - \Psi(x_i, y_i), \mathbf{w} \rangle \right).$$

We call the resulting method the *optimized cutting plane algorithm for multi-class SVM* (OCAM). Finally, note that the subtasks of OCAM can be parallelized in a fashion similar to the binary case (see Section 3.1.2).

3.2.1 LINE-SEARCH FOR GENERAL MULTI-CLASS LINEAR SVM CLASSIFIERS

In this section, we derive an efficient algorithm to solve the line-search (8) for the margin re-scaling risk (15). The algorithm is a generalization of the line-search for the binary SVM described in Section 3.1.1. Since the core idea remains the same we only briefly describe the main differences.

The goal of the line-search is to minimize a univariate function $F(\mathbf{w}_{t-1}^b(1-k) + \mathbf{w}_t k)$ defined by (13) with the risk R given by (15). We can abbreviate $F(\mathbf{w}_{t-1}^b(1-k) + \mathbf{w}_t k)$ by $f(k)$ which reads

$$f(k) := f_0(k) + \sum_{i=1}^{m} f_i(k) = \frac{1}{2} k^2 A_0 + k B_0 + C_0 + \sum_{i=1}^{m} \max_{y \in \mathcal{Y}} \left(k B_i^y + C_i^y \right),$$

where the constants $A_0, B_0, C_0, (B_i^y, C_i^y), i = 1, \ldots, m, y \in \mathcal{Y}$ are computed accordingly. Similar to the binary case, the core idea is to find an explicit formula for the subdifferential $\partial f(k)$, which, consequently, allows solving the optimality condition $0 \in \partial f(k)$. For a given $f_i(k)$, let $\hat{\mathcal{Y}}_i(k) = \{ y \in \mathcal{Y} \mid k B_i^y + C_i^y = \max_{y' \in \mathcal{Y}} \left(k B_i^{y'} + C_i^{y'} \right) \}$ be a set of indices of the linear terms which are active at the point k. Then the subdifferential of $f(k)$ reads

$$\partial f(k) = k A_0 + B_0 + \sum_{i=1}^{m} \partial f_i(k) \quad \text{where} \quad \partial f_i(k) = \operatorname{co}\{ B_i^y \mid y \in \hat{\mathcal{Y}}_i(k) \}. \tag{17}$$

The subdifferential (17) is composed of a linear term $kA_0 + B_0$ and a sum of maps $\partial f_i \colon \Re \to I$, $i = 1, \ldots, m$, where I is a set of all closed intervals on a real line. From the definition (17) it follows that ∂f_i is a step-function (or staircase function), that is, ∂f_i is composed of piece-wise linear horizontal and vertical segments. An explicit description of these linear segments is crucial for solving the optimality condition $0 \in \partial f(k)$ efficiently. Unlike the binary case, the segments cannot be computed directly from the parameters $(B_i^y, C_i^y), y \in \mathcal{Y}$, however, they can be found by the simple algorithm described below.

First, we introduce an equivalent representation of ∂f_i. Unlike (17), the new representation explicitly defines intervals where $\partial f_i(k)$ is a constant and the points for which the constant value of

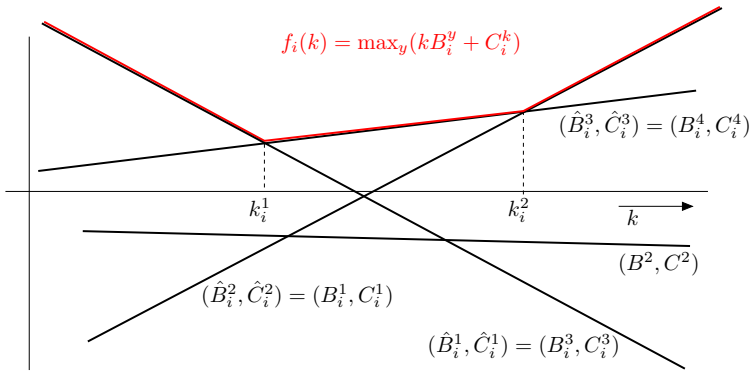


Figure 3: Figure shows an example of the function $f_i(k)$ which is defined as the point-wise maximum over linear terms $kB_i^y + C_i^y$, $y = 1,\ldots,4$. The parameters $(\hat{B}_i^z, \hat{C}_i^z)$, $z = 1,\ldots,3$, and points k_i^z, $z = 1,2$ found by Algorithm 3 are also visualized.

$\partial f_i(k)$ changes. Let $Z \in \{1,\ldots,Y-1\}$ be a given integer and $k_i^1,\ldots,k_i^{Z-1}$ be a strictly increasing sequence of real numbers. Then we define a system of Z open intervals $\{I_i^1,\ldots,I_i^Z\}$ such that

$$I_i^1 = (-\infty, k_i^1), \quad I_i^Z = (k_i^{Z-1}, \infty), \quad \text{and} \quad I_i^z = (k_i^{z-1}, k_i^z), \forall 1 < z < Z.$$

It can be seen that there exist an integer Z and a sequence $k_i^1,\ldots,k_i^{Z-1}$ such that the map ∂f_i can be equivalently written as

$$\partial f_i(k) = \begin{cases} \hat{B}_i^z & \text{if} \quad k \in I_i^z, \\ [\hat{B}_i^z, \hat{B}_i^{z+1}] & \text{if} \quad k \in k_i^z, \end{cases} \tag{18}$$

where $\{\hat{B}_i^1,\ldots,\hat{B}_i^Z\}$ is a subset of $\{B_i^1,\ldots,B_i^Y\}$. Provided the representation (18) is known for all ∂f_i, $i = 1,\ldots,m$, the line-search $k^* = \operatorname{argmin}_{k>0} f(k)$ can be solved exactly by finding the intersection of $\partial f(k)$ and the x-axis, that is, solving the optimality condition $0 \in \partial f(k)$. To this end, we can use the same algorithm as in the binary case (see Section 3.1.1). The only difference is that the number of points k_i^z in which the subdifferential $\partial f(k)$ changes its value is higher; $m \cdot (Y-1)$ in the worst case. As the computations of the algorithm for solving $0 \in \partial f(k)$ are dominated by sorting the points k_i^z, the worst-case computational complexity is approximately $O(m \cdot Y \log(m \cdot Y))$.

Finally, we introduce Algorithm 3, which finds the required representation (18) for a given ∂f_i. In the description of Algorithm 3, we do not use the subscript $_i$ to simplify the notation. Figure 3 shows an example of input linear terms (B_i^y, C_i^y), $y \in \mathcal{Y}$ defining the function $f_i(k)$ and the sorted sequence of active terms $(\hat{B}_i^z, \hat{C}_i^z)$, $z = 1,\ldots,Z$, and points k_i^z, $z = 1,\ldots,Z$, in which the activity of the linear terms changes. At the beginning, the algorithm finds a linear term which is active in the leftmost interval $(-\infty, k^1)$, that is, the line with the maximal slope. Then the algorithm computes intersections with the leftmost active linear term that was found and the remaining lines with lower slopes. The leftmost intersection identifies the next active term. This process is repeated until the rightmost active term is found. The worst-case computational complexity of Algorithm 3 is $O(Y^2)$. In turn, the total complexity of the line-search procedure is $O(m \cdot Y^2 + m \cdot Y \log(m \cdot Y))$, that is, $O(m \cdot Y^2)$ time is required for running Algorithm 3 m times and $O(m \cdot Y \log(m \cdot Y))$ time for solving the optimality condition $0 \in \partial f_i(k)$ as described above.

Algorithm 3 Finding explicit piece-wise linear representation (18) of ∂f_i

Require: $(B^y, C^y), y \in \mathcal{Y}$
Ensure: $Z, \{\hat{B}^1, \ldots, \hat{B}^Z\}$, and $\{k^1, \ldots, k^{Z-1}\}$
1: $\hat{y} := \text{argmax}_{y \in \hat{\mathcal{Y}}} C^y$ where $\hat{\mathcal{Y}} := \{y \mid B^y = \min_{y' \in \mathcal{Y}} B^{y'}\}$.
2: $Z := 1, k := -\infty$ and $\hat{B}^1 := B^{\hat{y}}$
3: **while** $k < \infty$ **do**
4: $\quad \hat{\mathcal{Y}} := \{y \mid B^y > B^{\hat{y}}\}$
5: $\quad$ **if** $\hat{\mathcal{Y}}$ is empty **then**
6: $\quad\quad k := \infty$
7: $\quad$ **else**
8: $\quad\quad \ddot{y} := \text{argmin}_{y \in \hat{\mathcal{Y}}} \frac{C^y - C^{\hat{y}}}{B^y - B^{\hat{y}}}$
9: $\quad\quad k^Z := \frac{C^{\ddot{y}} - C^{\hat{y}}}{B^{\ddot{y}} - B^{\hat{y}}}$
10: $\quad\quad Z := Z + 1$
11: $\quad\quad \hat{B}^Z := B^{\ddot{y}}$
12: $\quad\quad \hat{y} := \ddot{y}$
13: $\quad$ **end if**
14: **end while**

Note that the described algorithm is practical only if the output space $\mathcal{Y}$ is of moderate size since the complexity of the line-search grows quadratically with $Y = |\mathcal{Y}|$. For that reason, this algorithm is ineffective for structured output learning where the cardinality of $\mathcal{Y}$ grows exponentially with the number of hidden states.

4. Experiments

In this section we perform an extensive empirical evaluation of the proposed optimized cutting plane algorithm (OCA) applied to linear binary SVM classification (OCAS) and multi-class SVM classification (OCAM) .

In particular, we compare OCAS to various state-of-the-art SVM solvers. Since several of these solvers did not take part in the large-scale learning challenge, we perform an evaluation of SVM$^{\text{light}}$, Pegasos, GPDT, SGD, BMRM, SVM$^{\text{perf}}$ version 2.0 and version 2.1 on previously published medium-scale data sets (see Section 4.1.1). We show that OCAS outcompetes previous solvers gaining speedups of several orders of magnitude over some of the methods and we also analyze the speedups gained by parallelizing the various core components of OCAS.

In addition, we use the challenge data sets and follow the challenge protocol to compare OCAS with the best performing methods, which were LaRank and LibLinear (see Section 4.1.2). Finally, in section 4.2, we compare the multi-class SVM solver OCAM to the standard CPA implemented for multi-class SVM on four real-world problems using the challenge evaluation protocol.

4.1 Comparison of Linear Binary SVM

We first compare OCAS with several binary linear SVM solvers on previously published data sets followed by an analysis using the challenge criteria on the challenge data sets.

4.1.1 EVALUATION ON PREVIOUSLY USED DATA SETS

We now compare current state-of-the-art SVM solvers (SGD, Pegasos, SVM^{light} , SVM^{perf} , BMRM, GPDT[1]), on a variety of data sets with the proposed method (OCAS), using 6 experiments measuring:

1. Influence of the hyper-parameter μ on the speed of convergence
2. Training time and objective for optimal C
3. Speed of convergence (time vs. objective)
4. Time to perform a full model selection
5. Effects of parallelization
6. Scalability w.r.t. data set size

To this end, we implemented OCAS and the standard CPA[2] in C. We use the very general compressed sparse column (CSC) representation to store the data. Here, each element is represented by an index and a value (each 64bit). To solve the reduced problem (2), we use our implementation of improved SMO (Fan et al., 2005). The source code of OCAS is freely available for download as part of LIBOCAS (Franc and Sonnenburg, 2008b) and as a part of the SHOGUN toolbox (Sonnenburg and Rätsch, 2007).

All competing methods train SVM classifiers by solving the convex problem (1) either in its primal or dual formulation. Since in practice only limited precision solutions can be obtained, solvers must define an appropriate stopping condition. Based on the stopping condition, solvers can be categorized into *approximative* and *accurate*.

Approximative Solvers make use of heuristics (e.g., learning rate, number of iterations) to obtain (often crude) approximations of the optimal solution. They have a very low per-iteration cost and low total training time. Especially for large-scale problems, they are claimed to be sufficiently precise while delivering the best performance vs. training time trade-off (Bottou and Bousquet, 2007), which may be attributed to the robust nature of *large-margin* SVM solutions. However, while they are fast in the beginning they often fail to achieve a precise solution. Among the most efficient solvers to-date are Pegasos (Shwartz et al., 2007) and SGD (Bottou and Bousquet, 2007), both of which are based on stochastic (sub-)gradient descent.

Accurate Solvers In contrast to approximative solvers, accurate methods solve the optimization problem up to a given precision ε, where ε commonly denotes the violation of the relaxed KKT conditions (Joachims, 1999) or the (relative) duality gap. Accurate methods often have good asymptotic convergence properties, and thus for small ε converge to very precise solutions being limited only by numerical precision. Among the state-of-the-art accurate solvers are SVM^{light} , SVM^{perf} , BMRM and GPDT.

Because there is no widely accepted consensus on which approach is "better", we used both types of methods in our comparison.

1. Solvers include: SGD version 1.1 (svmsgd2) `http://leon.bottou.org/projects/sgd`, SVM^{light} 6.01 and SVM^{perf} 2.1 `http://svmlight.joachims.org`, pegasos `http://ttic.uchicago.edu/~shai/code/`, BMRM version 0.01 `http://users.rsise.anu.edu.au/~chteo/BMRM.html` and GPDT `http://dm.unife.it/gpdt`.
2. To not measure implementation specific effects (solver, dot-product computation etc.).

Experimental Setup We trained all methods on the data sets summarized in Table 2. We augmented the Cov1, CCAT, Astro data sets from Joachims (2006) by the MNIST, an artificial dense data set and two larger bioinformatics splice site recognition data sets for worm and human.[3]

Data Set	MNIST	Astro	Artificial	Cov1	CCAT	Worm	Human
Examples	70,000	99,757	150,000	581,012	804,414	1,026,036	15,028,326
Dim	784	62,369	500	54	47,236	804	564
Sparsity	19	0.08	100	22	0.16	25	25
Split	77/09/14	43/05/52	33/33/33	81/09/10	87/10/03	80/05/15	-

Table 2: Summary of the data sets used in the experimental evaluation. Sparsity denotes the average number of non-zero elements of a data set in percent. Split describes the size of the train/validation/test sets in percent.

These data sets have been used and are described in detail in Joachims (2006), Shwartz et al. (2007) and Franc and Sonnenburg (2008a). The Covertype, Astrophysics and CCAT data sets were provided to us by Shai Shalev-Shwartz and should match the ones used in Joachims (2006). The Worm splice data set was provided by Gunnar Rätsch. We did not apply any extra preprocessing to these data sets.[4]

The artificial data set was generated from two Gaussian distributions with different diagonal covariance matrices of multiple scale. Unless otherwise stated, experiments were performed on a 2.4GHz AMD Opteron Linux machine. We disabled the bias term in the comparison. As stopping conditions we use the defaults: $\varepsilon_{light} = \varepsilon_{gpdt} = 0.001$, $\varepsilon_{perf} = 0.1$ and $\varepsilon_{bmrm} = 0.001$. For OCAS we used the same stopping condition that is implemented in SVM$^{\text{perf}}$, that is, $\frac{F(\mathbf{w})-F_t(\mathbf{w})}{C} \leq \frac{\varepsilon_{perf}}{100} = 10^{-3}$. Note that these ε have very different meanings denoting the maximum KKT violation for SVM$^{\text{light}}$, the maximum tolerated violation of constraints for SVM$^{\text{perf}}$ and for BMRM the relative duality gap. For SGD we fix the number of iterations to 10 and for Pegasos we use $100/\lambda$, as suggested in Shwartz et al. (2007). For the regularization parameter C and λ we use the following relations: $\lambda = 1/C$, $C_{perf} = C/100$, $C_{bmrm} = C$ and $C_{light} = Cm$. Throughout the experiments we use C as a shortcut for C_{light}.[5]

Influence of the Hyper-parameter μ on the Speed of Convergence In contrast to the standard CPA, OCAS has a single hyper-parameter μ (see Section 3). The value of μ determines the point $\mathbf{w}_t^c = \mathbf{w}_t^b(1-\mu) + \mathbf{w}_t\mu$ at which the new cutting plane is selected. The convergence proof (see Theorem 2) requires μ to be from the interval $(0,1]$, however, the theorem does not indicate which value is the optimal one. For this reason, we empirically determined the value of μ.

For varying $\mu \in \{0.01, 0.05, 0.1, \ldots, 1\}$ we measured the time required by OCAS to train the classifier on the Astro, CCAT and Cov1 data sets. The regularization constant C was set to the

3. Data sets found at: Worm and Human `http://www.fml.tuebingen.mpg.de/raetsch/projects/lsmkl`, Cov1 `http://kdd.ics.uci.edu/databases/covertype/covertype.html`, CCAT `http://www.daviddlewis.com/resources/testcollections/rcv1/`, MNIST `http://yann.lecun.com/exdb/mnist/`.
4. However, we noted that the Covertype, Astro-ph and CCAT data set already underwent preprocessing because the latter two have $\|x_i\|_2 = 1$.
5. The exact cmdlines are: `svm_perf_learn -l 2 -m 0 -t 0 --b 0 -e 0.1 -c` C_{perf}, `pegasos -lambda` λ `-iter` $100/\lambda$ `-k 1`, `svm_learn -m 0 -t 0 -b 0 -e 1e-3 -c` C_{light}, `bmrm-train -r 1 -m 10000 -i 999999 -e 1e-3 -c` C_{bmrm}, `svmsgd2 -lambda` λ `-epochs 10`.

optimal value for the given data set. Figure 4 shows the results. For Astro and CCAT the optimal value is $\mu = 0.1$ while for Cov1 it is $\mu = 0.01$. For all three data sets the training time does not change significantly within the interval $(0, 0.2)$. Thus we selected $\mu = 0.1$ to be the best value and we used this setting in all remaining experiments.

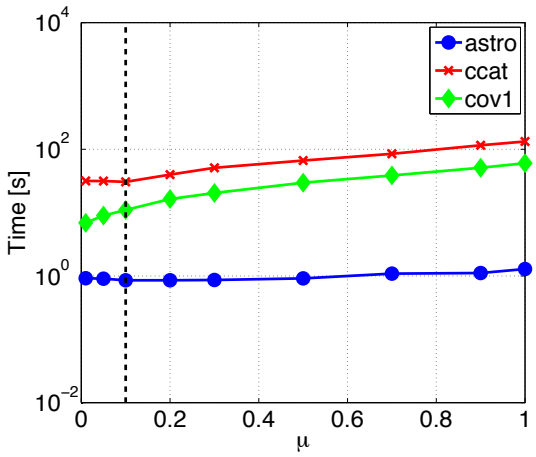


Figure 4: Training time vs. value of the hyper-parameter μ of the OCAS solver measured on the Astro, CCAT and Cov1 data sets. The value $\mu = 0.1$ (dash line) is used in all remaining experiments.

Training Time and Objective For Optimal C We trained all methods on all except the human splice data set using the training data and measured training time (in seconds) and computed the unconstrained objective value $F(\mathbf{w})$ (cf. Equation 11).

The results are displayed in Table 3. The proposed method—OCAS—consistently outperforms all its competitors in the *accurate solver* category on all benchmark data sets in terms of training time while obtaining a comparable (often the best) objective value. BMRM and SVM$^{\text{perf}}$ implement the same CPA algorithm but due to implementation-specific details, results can be different. Our implementation of CPA gives very similar results (not shown).[6] Note that for SGD, Pegasos (and SVM$^{perf2.0}$—not shown), the objective value sometimes deviates significantly from the true objective. As a result, the learned classifier may differ substantially from the optimal parameter $\mathbf{w}^*$. However, as training times for SGD are significantly below all others, it is unclear whether SGD achieves the same precision using less time with further iterations. An answer to this question is given in the next paragraph.

Speed of Convergence (Time vs. Objective) To address this problem we re-ran the best methods, CPA, OCAS and SGD, recording *intermediate* progress, that is, in the course of optimization record time and objective for several time points. The results are shown in Figure 5. OCAS was stopped when reaching the maximum time or a precision of $1 - F(\mathbf{w}^*)/F(\mathbf{w}) \leq 10^{-6}$ and in all cases achieved the minimum objective. In three of the six data sets, OCAS not only, achieves the

6. In contrast to SVM$^{\text{perf}}$, BMRM and our implementation of CPA did not converge for large C on Worm even after 5000 iterations. Most likely, the core solver of SVM$^{\text{perf}}$ is more robust.

	Astro		CCAT		Cov1		MNIST		Worm		Artificial	
svmlight	**2.0939e+03**		**8.1235e+04**		**2.5044e+06**		**6.7118e+05**		**3.1881e+04**		**1.3170e+02**	
	2972	22	77429	5295	1027310	41531	622391	2719	2623193	44852	231059	3060
svmperf2.1	**2.1180e+03**		**8.1744e+04**		**2.5063e+06**		**6.7245e+05**		**3.2224e+04**		**1.3186e+02**	
	38	2	228	228	520	152	1295	228	2029	4436	709	162
svmperf2.0	**2.1188e+03**		**8.1760e+04**		**2.5071e+06**		**6.7276e+05**		**3.2327e+04**		**1.3182e+02**	
	-1	11	-1	1250	-1	345	-1	6115	-1	16515	-1	455
bmrm	**2.1152e+03**		**8.1682e+04**		**2.5060e+06**		**6.7250e+05**					
	42	2	327	248	678	225	2318	4327		-		
ocas	**2.1103e+03**		**8.1462e+04**		**2.5045e+06**		**6.7158e+05**		**3.1920e+04**		**1.3172e+02**	
	21	1	48	25	80	10	137	10	125	258	76	13
pegasos	**2.1090e+03**		**8.1564e+04**		**2.5060e+06**		**Error**		**4.6212e+04**		**1.3120e+03**	
	2689K	4	70M	127	470M	460	270M	647	82M	213	25K	1
sgd	**2.2377e+03**		**8.2963e+04**		**2.6490e+06**		**1.3254e+06**		**2.1299e+05**		**1.8097e+02**	
	10	1	10	4	10	1	10	1	10	9	10	2
gpdt	**1.1725e+03**		**1.5418e+05**		**1.3034e+06**		**5.9796e+06**		**1.3205e+04**		**1.2642e+02**	
	130	5	3570	2263	4844	1794	526	118	38092	39095	615	137

Table 3: Training time for optimal C comparing OCAS with other SVM solvers. "-" means not converged, blank not attempted. Shown in bold is the unconstrained SVM objective value Eq. (11). The two numbers below the objective value denote the number of iterations (left) and the training time in seconds (right). Lower time and objective values are better. All methods solve the unbiased problem. As convergence criteria, the standard settings described in Section 4.1.1 are used. On MNIST Pegasos ran into numerical problems. OCAS clearly outperforms all of its competitors in the *accurate solver* category by a large margin achieving similar and often the lowest objective value. The objective value obtained by SGD and Pegasos is often far away from the optimal solution; see text for a further discussion.

best objective as expected at a later time point, but already from the very beginning. Further analysis made clear that OCAS wins over SGD in cases where *large C*s were used and thus the optimization problem is more difficult. Still, plain SGD outcompetes even CPA. One may argue that, practically, the true objective is not the unconstrained SVM-primal value (11) but the performance on a validation set, that is, optimization is stopped when the validation error does not change. This has been discussed for leave-one-out in Franc et al. (2008) and we—to some extent—agree with this. One should, however, note that in this case one does not obtain an SVM but *some classifier* instead. A comparison should not then be limited to SVM solvers but should also be open to any other large scale approach, like online algorithms (e.g., perceptrons). We argue that to compare *SVM solvers* in a fair way one needs to compare objective values. We therefore ran Pegasos using a larger number of iterations on the Astro and splice data sets. On the Astro data set, Pegasos surpassed the SVM$^{\text{light}}$ objective after 10^8 iterations, requiring 228 seconds. SVM$^{\text{light}}$, in comparison, needed only 22 seconds. Also, on the splice data set we ran Pegasos for 10^{10} iterations, which took 13,000 seconds and achieved a similar objective as that of SVM$^{perf2.0}$, requiring only 1224 seconds. Finally, note that, although BMRM, SVM$^{\text{perf}}$ and our implementation of CPA solve the same equivalent problem using the CPA, differences in implementations lead to varying results.[7] Since it

7. Potentially due to a programming error in this pre-release of BMRM, it did not show convergence on the splice data set even after > 6500 iterations.

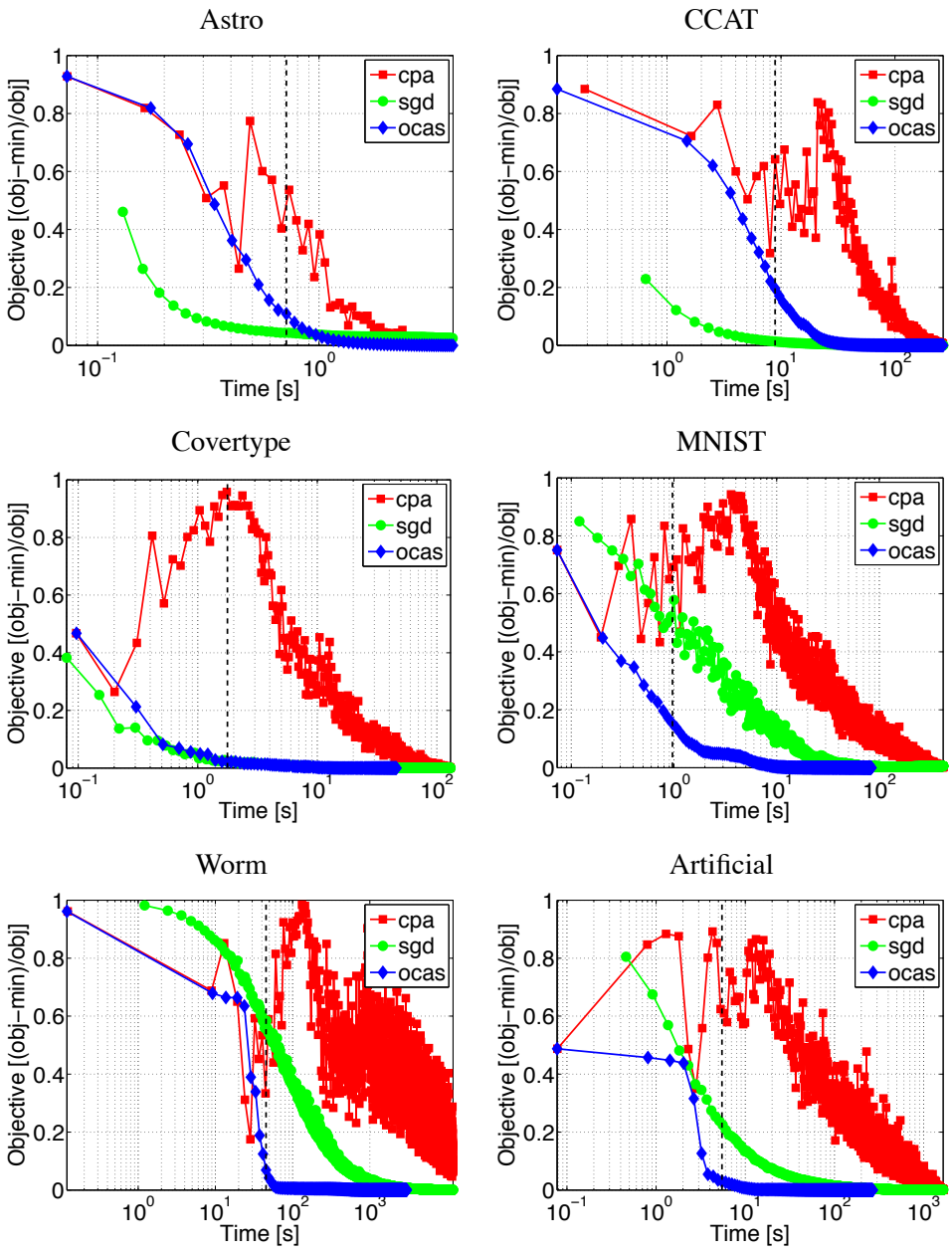


Figure 5: Objective value vs. training time of CPA (red), SGD (green) and OCAS (blue) measured for different numbers of training examples.The dashed line shows the time required to run SGD for 10 iterations. OCAS was stopped when the precision fell below 10^{-6} or the training time for CPA was achieved. In all cases, OCAS achieves the minimal objective value and, even from the beginning, outperforms all other methods, including SGD, on half of the data sets.

is still interesting to see how the methods perform w.r.t. classification performance, we describe the analysis under this criterion in the next paragraph.

Time to Perform a Full Model Selection When using SVMs in practice, their C parameter needs to be tuned during model selection. We therefore train all methods using different settings[8] for C on the training part of all data sets, evaluate them on the validation set and choose the best model to do predictions on the test set. As the performance measure, we use the area under the receiver operator characteristic curve (auROC) (Fawcett, 2003). Results are displayed in Table 4.

	Astro		CCAT		Cov1		MNIST		Worm		Artificial	
avg svm perf	**98.15 ± 0.00**		**98.51 ± 0.01**		**83.92 ± 0.01**		**95.86 ± 0.01**		**99.45 ± 0.00**		**86.38 ± 0.02**	
svmlight	1	152	1	124700	10	282703	10	9247	0.5	86694	0.005	42491
svmperf2.0	1	67	1	20827	50	1765	5	21113	5	106241	0.005	111621
svmperf2.1	1	13	1	1750	5	781	10	887	1	22983	0.005	24520
bmrm	1	17	1	2735	10	1562	10	20278	-			
ocas	1	4	1	163	50	51	10	35	0.1	1438	0.005	6740
pegasos	**98.15**		**98.51**		**83.89**		**95.84**		**99.27**		**78.35**	
	1	59	1	2031	5	731	5	2125	5	1438	5	201
sgd	**98.13**		**98.52**		**83.88**		**95.71**		**99.43**		**80.88**	
	0.5	1	1	20	1	5	1	3	0.005	69	0.005	7
gpdt	1	30	1	33693	5	11615	10	408	0.5	283941	0.005	90807

Table 4: Model selection experiment comparing OCAS with other SVM solvers. "-" means not converged, blank not attempted. Shown in bold is the area under the receiver operator characteristic curve (auROC) obtained for the best model chosen based on model selection over a wide range of regularization constants C. In each cell, numbers on the left denote the optimal C, numbers on the right the training time in seconds to perform the whole model selection. Because there is little variance, for accurate SVM solvers only the mean and standard deviation of the auROC are shown. SGD is clearly fastest achieving similar performance for all except the artificial data set. However, often a C smaller than the ones chosen by accurate SVMs is selected—an indication that the learned decision function is only remotely SVM-like. Among the accurate solvers, OCAS clearly outperforms its competitors. It should be noted that training times for all accurate methods are dominated by training for large C (see Table 3 for training times for the optimal C). For further discussion see the text.

Again, among the *accurate methods* OCAS outperforms its competitors by a large margin, followed by SVM$^{\text{perf}}$. Note that for all *accurate methods* the performance is very similar and has little variance. Except for the artificial data set, plain SGD is clearly fastest while achieving a similar accuracy. However, the optimal parameter settings for accurate SVMs and SGD are different. Accurate SVM solvers use a larger C constant than SGD. For a lower C, the objective function is dominated by the regularization term $\|w\|$. A potential explanation is that SGD's update rule puts more emphasis on the regularization term, and SGD, when not run for a large number of iterations, does imply early stopping.

Our suggestion for practitioners is to use OCAS whenever a reliable and efficient large-scale solver with proven convergence guarantees is required. This is typically the case when the solver is

8. For Worm and Artificial we used $C \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5\}$, for CCAT, Astro, Cov1 $C \in \{0.1, 0.5, 1, 5, 10\}$ and for MNIST $C \in \{1, 5, 10, 50, 100\}$.

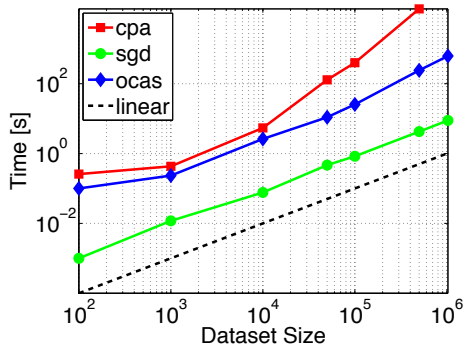


Figure 6: This figure displays how the methods scale with data set size on the Worm splice data set. The slope of the "lines" in this figure denotes the exponent e in $O(m^e)$, where the black line denotes linear effort $O(m)$.

to be operated by non-expert users who know little (or nothing) about tuning the hyper-parameters of the optimization algorithm. Therefore, as long as the full data set fits into memory, we recommend OCAS. Otherwise, if sub-sampling is not an option, online approximative solvers like SGD are the only viable way to proceed.

Effects of Parallelization As OCAS training times are very low on the above data sets, we also apply OCAS to the 15 million human splice data set. Using a 2.4GHz 16-core AMD Opteron Linux machine, we run OCAS using $C = 0.0001$ on 1 to 16 CPUs and show the accumulated times for each of the subtasks, the total training time and the speedup w.r.t. the single CPU algorithm in Table 5. Also shown is the accumulated time for each of the threads. As can be seen—except for the line-

CPUs	1	2	4	8	16
speedup	1	1.77	3.09	4.5	4.6
line search (s)	238	184	178	139	117
$\mathbf{a}_t$ (s)	270	155	80	49	45
output (s)	2476	1300	640	397	410
total (s)	3087	1742	998	684	671

Table 5: Speedups due to parallelizing OCAS on the 15 million human splice data set.

search—computations distribute nicely. Using 8 CPU cores the speedup saturates at a factor of 4.5, most likely as memory access becomes the bottleneck (for 8 CPUs output computation creates a load of 28GB/s just on memory reads).

Scalability w.r.t. Data Set Size In this section, we investigate how computational times of OCAS, CPA and SGD scale with the number of examples on the Worm splice data set for sizes 100 to 1,026,036. Results are shown in Figure 6. We again use our implementation of CPA which shares essential sub-routines with OCAS. Both OCAS and SGD scale roughly linearly. Note that SGD is much faster (because it runs for a fixed number of iterations and thus stops early).

4.1.2 EVALUATION ON CHALLENGE DATA

In this section, we use the challenge data sets and follow the challenge protocol to compare OCAS to the best-performing methods, which were LaRank (Bordes et al., 2007) and LibLinear (Fan et al., 2008). To this end, we apply OCAS to the challenge data sets Alpha, Gamma and Zeta following the challenge protocol for the SVM track.

The data sets are artificially generated based on a mixture of Gaussians and have certain properties (see Table 6): The Alpha data set is separable with a large margin using quadratic features.

Data Set	Optimal Model	Number of examples training	testing	validation	Dim.	Description
Alpha	quadratic	500,000	300,000	100,000	500	well separable
Gamma	semi-quadratic	500,000	300,000	100,000	500	Multiscale low var.
Zeta	linear	500,000	300,000	100,000	2000	Intrinsic dim. 400

Table 6: Summary of the three challenge data sets used: Alpha, Gamma, Zeta.

The Gamma data set is well separable too, but contains features living on different scales. Finally, the optimal model for Zeta is a linear classifier—of its 2,000 features 1,600 are nuisance dimensions. The challenge protocol requires training on the unmodified data sets with $C = 0.01$ and precision $\varepsilon = 0.01$. To measure convergence speed, objective values are measured *while training*. The second challenge experiment simulates model selection by training for different $C \in \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$.

The left column in Figure 7 displays the course of convergence of the three methods. While OCAS is quite competitive on Gamma and Zeta in this experiment, it is slower on Alpha. It should also be noted that OCAS, in contrast to the online-style algorithms LaRank and LibLinear, has to do a full pass through the data in each iteration. However, it usually requires very few iterations to obtain precise solutions.

In the simulated model selection experiment (right column of Figure 7), OCAS performs well for low values of C on all data sets. However, at first glance it is competitive for large values of C only on Zeta. Investigating objective values on Gamma for LibLinear, we noticed that they significantly deviate (objective values much larger, deviation by 50% for $C = 10$) from LaRank/OCAS for $C \in \{1, 10\}$. Still, on Alpha OCAS is slower.

4.2 Comparison of Linear Multi-Class SVMs

In this section, we compare the proposed multi-class SVM solver OCAM described in Section 3.2 with multi-class CPA (CPAM). We consider the Crammer and Singer (2001) formulation of multi-class SVMs which corresponds to the minimization of the following convex objective,

$$F(\mathbf{w}) := \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m} \max_{y \in \mathcal{Y}} \left([\![y_i \neq y]\!] + \langle \mathbf{w}_y - \mathbf{w}_{y_i}, \mathbf{x}_i \rangle \right), \tag{19}$$

where $\mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_Y]$ is a matrix of parameter vectors and $\{(\mathbf{x}_1, y_i), \ldots, (\mathbf{x}_m, y_m)\} \in (\Re^n \times \mathcal{Y})^m$ is a set of training examples. The multi-class classification rule then reads $h(\mathbf{x}) = \operatorname{argmax}_{y \in \mathcal{Y}} \langle \mathbf{x}, \mathbf{w}_y \rangle$.

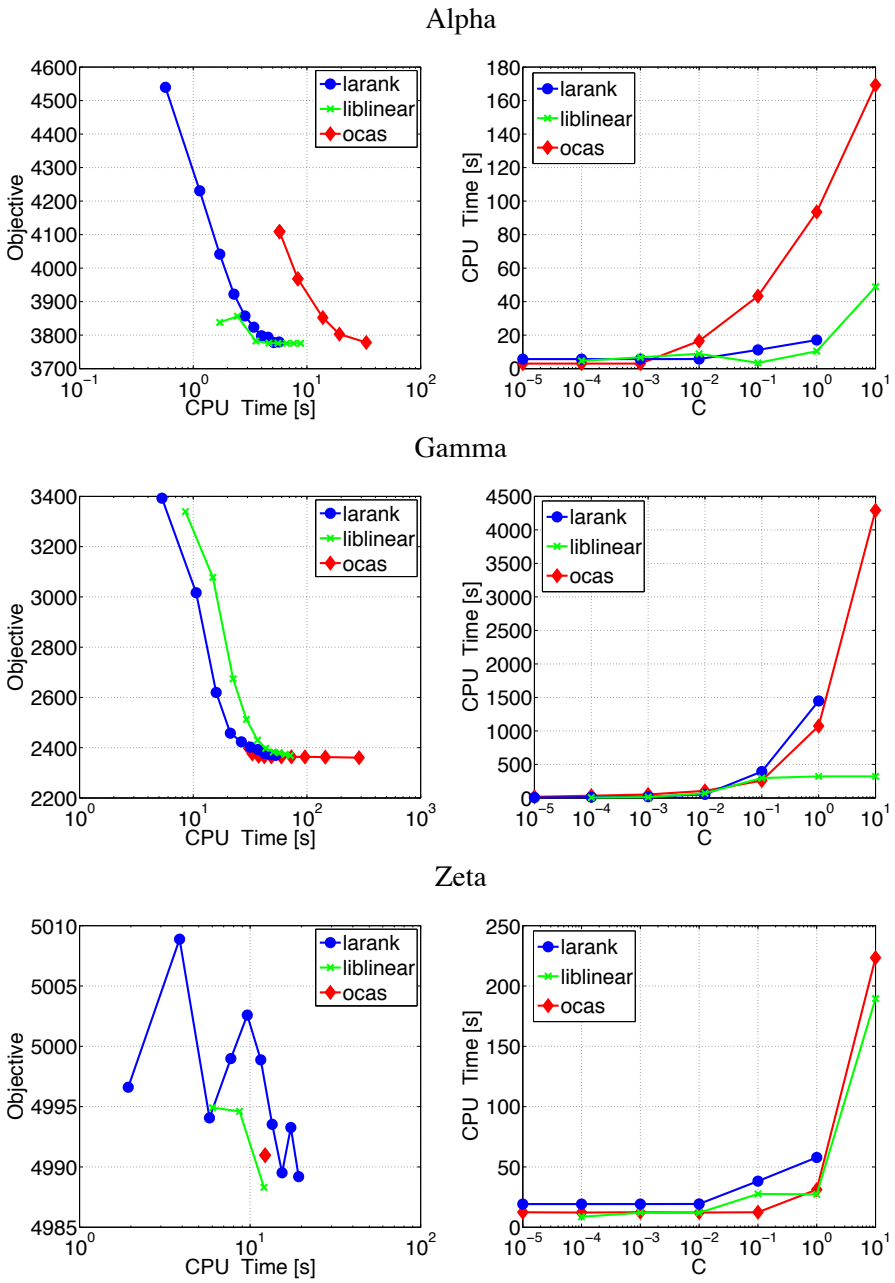


Figure 7: Results of LaRank, LibLinear and OCAS on the Alpha, Gamma and Zeta challenge data sets. The left column of the figures displays the unconstrained SVM primal objective (11) (C is not scaled with m) w.r.t. SVM training time for fixed $C = 0.01$. The right column displays the SVM training time for different C. We omitted the data set size vs. CPU time figure since all three methods show a similar curve (a line with slope ≈ 1 in log-log representation, corresponding to the expected $O(m)$ effort) . Note that for the Zeta data set OCAS converges after a single pass through the data, which results in collapsing the performance curve into a single point. For low C, OCAS achieves very competitive results. For further explanation see text.

We implemented OCAM and CPAM in C, exactly according to the description in Section 3.2. Both implementations use the Improved Mitchel-Demyanov-Malozemov algorithm (Franc, 2005) as the core QP solver and they use exactly the same functions for the computation of cutting planes and classifier outputs. The implementation of both methods is freely available for download as part of LIBOCAS (Franc and Sonnenburg, 2008b). The experiments are performed on an AMD Opteron-based 2.2GHz machine running Linux.

In the evaluation we compare OCAM with CPAM to minimize programming bias. In addition, we perform a comparison with SVM$^{\text{multi-class}}$ v2.20 later in Section 5.2.

We use four data sets with inherently different properties that are summarized in Table 7. The Malware data set is described in Section 5.2. The remaining data sets, MNIST, News20 and Sector, are downloaded from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`. We used the versions with the input features scaled to the interval $[0,1]$. Each data set is randomly split into a training and a testing part.

	features		number of	num. of examples	
	number of	type	classes	training	testing
Malware	3,413	dense	14	3,413	3,414
MNIST	780	dense	10	60,000	10,000
News20	62,060	sparse	20	15,935	3,993
Sector	55,197	sparse	105	6,412	3,207

Table 7: Multi-class data sets used in the comparison of OCAM and CPAM.

	Malware		MNIST		News20		Sector	
	error	time	error	time	error	time	error	time
Standard CPAM	10.25	12685	7.07	15898	14.45	7276	5.58	12840
Proposed OCAM	10.16	1705	7.08	5387	14.45	1499	5.61	3970
speedup		7.4		3.0		4.9		3.2

Table 8: Comparison of OCAM and CPA on a simulated model selection problem. The reported time corresponds to training over the whole range of regularization constants Cs. The error is the minimal test classification over the classifiers trained with different Cs.

In the first experiment, we train the multi-class classifiers on training data with a range of regularization constants $C = \{10^0, 10^1, \ldots, 10^7\}$ (for Malware $C = \{10^0, \ldots, 10^8\}$ since the optimal $C = 10^7$ is the boundary value). Both solvers use the same stopping condition (10) with $\varepsilon = 0.01F(\mathbf{w})$. We measure the total time required for training over the whole range of Cs and the best classification error measured on the testing data. Table 8 summarizes the results. While the classification accuracy of OCAM and CPAM are comparable, OCAM consistently outperforms the standard CPAM in terms of runtime, achieving speedup of factor from 3 to 7.4.

In the second experiment, we measure the three performance figures defined in the large-scale challenge: (i) the objective value as a function of the runtime, (ii) the runtime as a function of C and (iii) the runtime as a function of the data set size. For figures (i) and (ii) we use the optimal C obtained in the first experiments. Results for the first three data sets are shown in Figure 8.

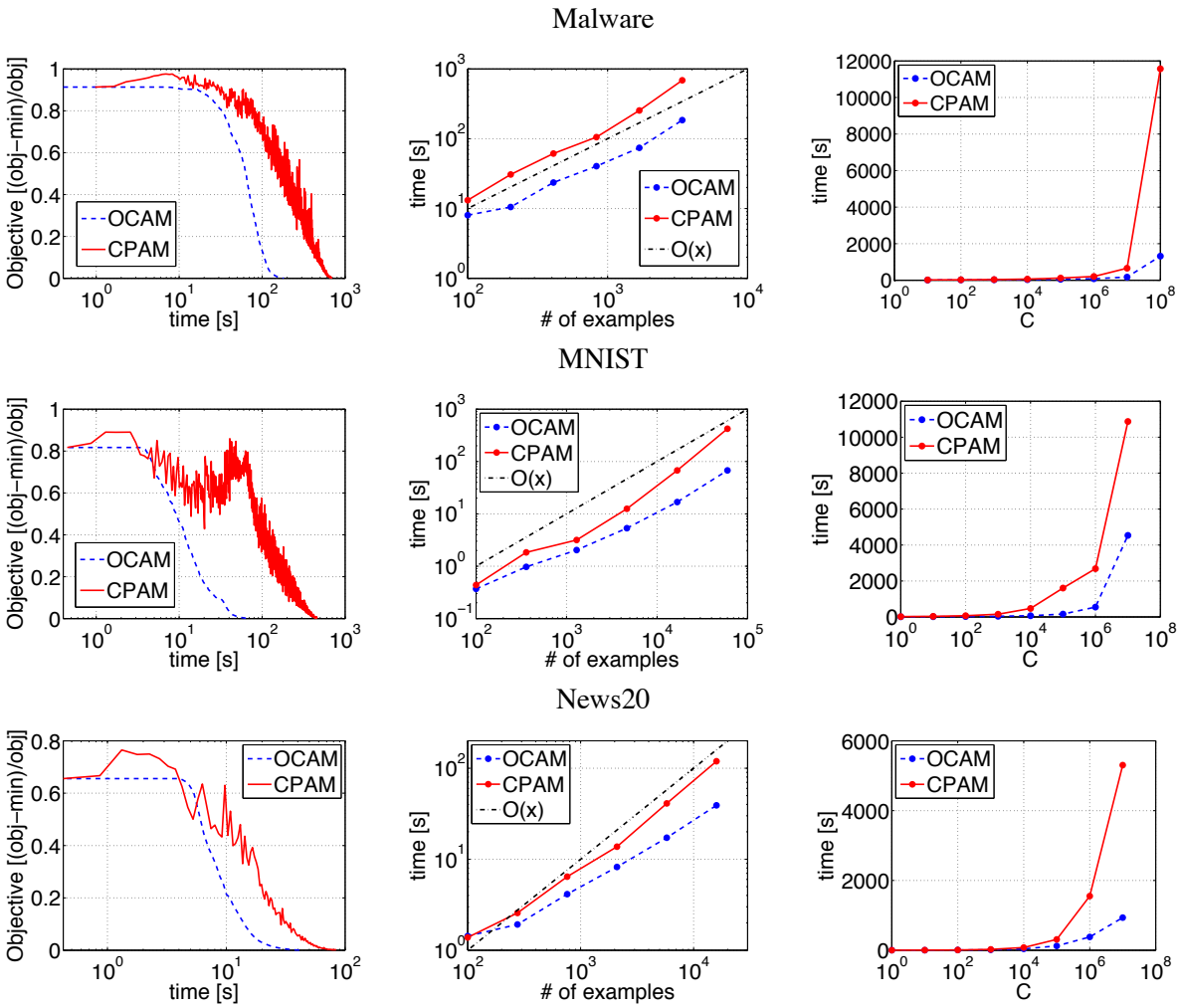


Figure 8: Results of the standard CPAM and the proposed OCAM on the Malware, MNIST and News20 data sets. The left column of figures displays the unconstrained SVM objective (19) w.r.t. SVM training time. The middle column displays the training time as a function of the number of examples. In both experiments C was fixed to its optimal value as determined in model selection. The right column shows the training time for different Cs. See text for a discussion of the results.

The objective vs. time figure is consistent with the results obtained in Section 4.1 for the two-class variant, that is, the objective value of the standard CPAM significantly fluctuates while OCAM decreases the objective monotonically and converges faster in all cases. The data size vs. time figure shows that in both cases the runtime is approximately linear w.r.t. the number of examples, though the curve of CPAM grows slightly faster compared to OCAM. The main difference shows the figure depicting the runtime as a function of C. It is seen that OCAM is considerably faster for large values of C, which is crucial for efficient model selection (see the experiment in Section 5.2) .

5. Applications

In this section we attack two real-world applications. First, in Section 5.1, we apply OCAS to a human acceptor splice site recognition problem. Second, in Section 5.2, we use OCAM for learning a behavioral malware classifier.

5.1 Human Acceptor Splice Site Recognition

To demonstrate the effectiveness of our proposed method, OCAS, we apply it to the problem of human acceptor splice site detection. Splice sites mark the boundaries between potentially protein-coding exons and (non-coding) introns. In the process of translating DNA to protein, introns are excised from pre-mRNA after transcription (Figure 9). Most splice sites are so-called *canonical splice sites* that are characterized by the presence of the dimers `GT` and `AG` at the donor and acceptor sites, respectively.

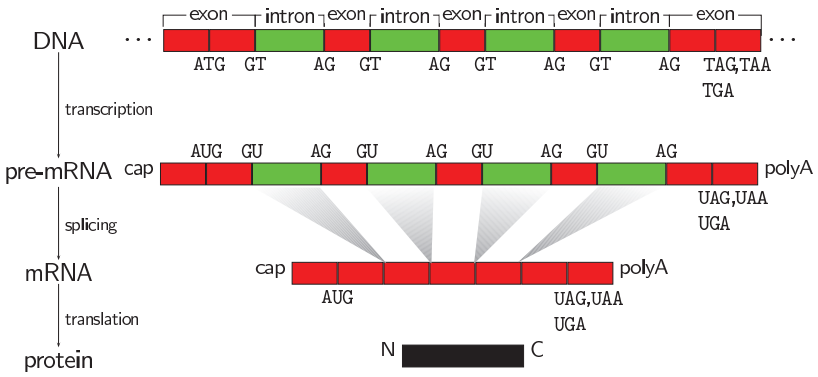


Figure 9: The major steps in protein synthesis. In the process of converting DNA to messenger RNA, the introns (green) are spliced out. Here we focus on detecting the so-called acceptor splice sites that employ the `AG` consensus and are found at the "left-hand side" boundary of exons. Figure taken from (Sonnenburg, 2002).

However, the occurrence of the dimer alone is not sufficient to detect a splice site. The classification task for splice site sensors, therefore, consists in discriminating true splice sites from decoy sites that also exhibit the consensus dimers. Assuming a uniform distribution of the four bases, adenine (`A`), cytosine (`C`), guanine (`G`) and thymine (`T`), one would expect 1/16th of the dimers to contain the `AG` acceptor splice site consensus. Considering the size of the human genome, which consists of about 3 billion base pairs, this constitutes a large-scale learning problem (the expected number of `AG`s is 180 million).

Many different methods to detect splice sites have been proposed. They all predict splice sites based on the local context, that is, a short window around the `AG` dimer. Currently, support vector machines are the most accurate splice site detectors (Degroeve et al., 2005; Sonnenburg et al., 2007b). Sonnenburg et al. (2007b) showed that prediction accuracy steadily increases with training sample size. However, even though they already used the `linadd` algorithm (Sonnenburg et al., 2007a) to speed up string kernel-based SVMs on a quad-core system, they could not use all available 50 million training points (but "only" 8 million). The string kernel that performed best was the

weighted degree (WD) string kernel *with shifts* (Rätsch et al., 2005). It basically counts matching k-mers for various k in a position-dependent way. Employing a giant string kernel feature space, Sonnenburg et al. (2007b) achieved $45.58\% \pm 0.38$ aoPRC in a genome-wide study on human acceptor splice sites—also available as the DNA data set used in the large-scale learning challenge.

On the other hand, Degroeve et al. (2005) trained a linear SVM based on a number of preselected and explicitly computed string kernel feature spaces that are subsets from the spectrum (Leslie et al., 2002) and WD kernel (Rätsch et al., 2005) feature spaces: Left and right of the splice site spectrum kernels of order 3 up to order 6 were used (Leslie et al., 2002). Over the whole window, a WD kernel of order 3 with weights equal to 1 was used (Rätsch and Sonnenburg, 2004). Even though this approach scales well, they used $< 100,000$ data points (potentially, since they relied on the unmodified SVM$^{\text{light}}$ binary).

Here, we propose to train OCAS on all available 50 million strings of length 141 from Sonnenburg et al. (2007b) using the features corresponding to two weighted spectrum kernels (one left and one right of the splice site, that is, positions 1-59 and 62-141) and a WD kernel (applied to the whole string). For the spectrum kernels of order 1 up to 8 and for the WD kernel of order 8 is used. Thus, the spanned string kernel feature space has 12,495,340 dimensions.

As the raw string-based data set already has a size of $7.1 \cdot 10^9$ bytes and even a sparse representation of each string would increase the data set by a factor of more than 3,000 ($(141+59+80) \cdot 12$ bytes per feature vector, assuming a 4 byte integer and an 8 byte float), we will *implicitly* compute features from the raw input strings on demand. The only required operations in OCAS for which we will have to expand the features are the addition to a dense vector $\mathbf{w} \leftarrow \mathbf{w} + \alpha\Phi(x)$ and the output computation $\mathbf{w} \cdot \Phi(x)$.

We implemented a rather general framework that allows stacking of arbitrary features that support such operations (dense and sparse real-valued, weighted spectrum and WD kernel features for specified k-mer length). As we know from Section 4.1.1, most time is spent in computing outputs, hence we parallelized this part of the code (based on shared memory parallelization, that is, posix threads).

Before training on the 50 million examples, we perform model selection on only 1 million examples to determine the optimal k-mer length for the two spectrum kernels, the WD kernel and its weighting and the SVM regularization constant C. The optimal parameter setting was found to be $C = 1$, $k_{wspec} = 8$, $k_{wd} = 8$, where the WD kernel weights are taken from the first 8 weights of the weights of a wd kernel of order 40.[9] Parameters were selected from $C \in \{0.5, 1, 3, 5, 10\}$, $k_{wspec} \in \{3, 6, 8\}$, $k_{wd} = \{3, 6, 8\}$ with the WD kernel-weighting from order 8, 25 or 40.

We then trained on 50 million examples on an 8-core AMD Opteron Linux-based machine, obtaining a record area over the precision recall curve (aoPRC) of 42.23%. For comparison, the previous best method achieved aoPRC of 45.58% (variance 0.38%). Note that this is the DNA data set used in the large-scale learning challenge, for which the best participant obtained an aoPRC of 80.89% (lower is better). OCAS converged in just 138 iterations, however, the total training time was about 40 hours, of which almost 34 hours were spent on computing outputs (already in a parallelized way; see Table 9 for the detailed timing statistics). Even though we observed that this parallelization was quite effective, it suggests that we are measuring random memory access speed. Due to the size of the normal vector (about 100MB since we are using double precision floats) we see only cache misses. This suggests that even using just single precision floats would reduce the

9. In Sonnenburg (2008) it was suggested that the WD kernel-weighting influences the effect of mismatches of the WD kernel score.

training time by 17 hours. Even though modern DDR-SDRAM is capable of speeds of up to 8 GB/s (Wikipedia, 2009) when being accessed in a linear way, we observed a memory speed of only 1.4GB/s on this system. It turns out that only DDR-333 memory is installed with a peak transfer rate of 2.7GB/s. Thus, additional speedups can be achieved by distributed memory parallelization and by grouping the access of features in $\mathbf{w}$ to minimize cache misses. Alternatively, switching to a many-core architecture like the NVIDIA Tesla s1070 computing system[10] that employs 960 CPU cores and a peak memory rate of 400GB/s could drastically reduce training times, potentially to even under 1 minute. Finally, it should be noted that storing the 138 cutting planes required almost 13 GB of memory.

Iterations	Output	Line Search	Add $\mathbf{a}_t$	Solver	Total
138	34 hours	222s	7 hours	5min	41 hours

Table 9: Timing statistics for the human acceptor splice site experiment.

5.2 Malware Classification

Malware is malicious software that occurs in the form of Internet worms, computer viruses and Trojan horses. Due to an enormous increase of new variants of malware, methods for its automatic detection and categorization are becoming crucial in modern anti-malware products. Rieck et al. (2008) propose a malware behavioral classifier trained from labeled examples. Malware binaries are collected via honeypots and spam-traps, and malware family labels are generated by running an anti-virus tool. This results in a corpus of more than 10,000 unique malware instances. The behavior of each binary is monitored in a sand box environment and behavior-based analysis reports summarizing operations, such as opening an outgoing IRC connection or stopping a network service, are generated. The reports have a form of text files which are then embedded into a high-dimensional vector space using the bag-of-words model. Finally, a discriminative multi-class SVM classifier is trained.

Rieck et al. (2008) use the multi-class classifier based on one-against-all decomposition, where each binary classifier is trained by a kernel SVM. To increase classification performance, the scale of the independently trained binary discriminant functions, forming the multi-class classifier, is normalized by fitting a logistic function. Rieck et al. (2008) report promising results achieving 88% classification accuracy, which is competitive with commercial anti-virus software tools handcrafted manually by computer security experts. Apart from the classification accuracy, the ability to retrain swiftly on new examples is a crucial feature for practical application of the system. While the classification accuracy was the main focus in Rieck et al. (2008), the issue of fast training was not addressed. The SVM$^{\text{light}}$ that they used required approximately 13-14 hours to perform the whole model selection on a high-end single CPU computer.[11]

To resolve the problem of fast training, we apply our proposed OCAM solver and compare its performance with SVM$^{\text{multi-class}}$ (Joachims et al., 2009). SVM$^{\text{multi-class}}$ version 2.20[12] is a highly optimized implementation of CPAM which uses numerous heuristic speedups like adaptive accu-

10. Found at `http://www.nvidia.com/object/product_tesla_s1070_us.html`.
11. Personal communication with authors of Rieck et al. (2008).
12. Found at `http://www.cs.cornell.edu/People/tj/svm_light/svm_multiclass.html`.

racy management, caching or 1-slack reformulation (for more details see Joachims et al. 2009). Note that OCAM is the plain implementation of the proposed Algorithm 2, hence there is still the possibility to improve its performance by implementing the same heuristics. SVM$^{\text{multi-class}}$ optimizes a slightly modified risk $R'(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \max_{y\in\mathcal{Y}} \left(100[\![y_i \neq y]\!] + \langle \mathbf{w}_y - \mathbf{w}_{y_i}, \mathbf{x}_i\rangle\right)$. To make objectives of SVM$^{\text{multi-class}}$ and OCAM equivalent we use the transform: $\mathbf{x}_i = \mathbf{x}'_i/100$ and $C = 100C'$ where $\mathbf{x}'_i$ and C' denote inputs and the regularization constant used by SVM$^{\text{multi-class}}$. SVM$^{\text{multi-class}}$ stops optimization when $F(\mathbf{w}_t) - F_t(\mathbf{w}_t) \leq C\varepsilon'$, hence we apply $\varepsilon = C\varepsilon'$ in OCAM to use equivalent stopping conditions. In addition, we use $\varepsilon' = 0.1$, which is the default setting in SVM$^{\text{multi-class}}$.

solver	error [%]	training time	
SVM$^{\text{multi-class}}$ v2.20	11.45 ± 0.72	25,330 sec	$\approx$ 7 hours
OCAM	11.49 ± 0.91	2,451 sec	$\approx$ 40 minutes
Rieck et al. (2008)	12		$\approx$ 13-14 hours

Table 10: Comparison of SVM$^{\text{multi-class}}$ v2.20 with the proposed OCAM on the malware classification problem. The reported error is a 5-fold cross-validation estimate of the per-class average classification error. The training time refers to the total time required by model selection. We also compare with the previous results reported in Rieck et al. (2008). OCAS achieves a speedup of factor 10 over SVM$^{\text{multi-class}}$ and of factor 20 compared to the one-against-all based classifier trained by kernel SVM$^{\text{light}}$ used in Rieck et al. (2008).

We adopted the evaluation protocol from Rieck et al. (2008). The classification accuracy is measured in terms of average per-class classification error, that is, $E = \frac{1}{Y}\sum_{y\in\mathcal{Y}} \frac{1}{m_i}\sum_{i|y_i=y}[\![y_i \neq h(\mathbf{x}_i)]\!]$, where m_i is the number of examples in the i-th class. The malware corpus of 10,072 examples is randomly split 5 times into training, validation and testing partitions of approximately equal size. The training partition is used to train the multi-class SVM using a range of different regularization constants $C \in \{10^0, 10^1, \ldots, 10^{10}\}$. The best C is selected based on classification accuracy measured on the validation part. Finally, we report average and standard deviations of the per-class average classification error computed on testing data over the 5 random splits.

Due to the very high-dimensional feature space, the explicit representation of the input vectors is inefficient. To apply the linear SVM solvers, we use the standard trick of representing the kernel matrix by the whitened empirical kernel map (Schölkopf and Smola, 2002). This representation reduces the number of features to the number of training examples. The runtime required by the singular value decomposition (SVD) to compute the whitened matrix is approximately 5 minutes, which is negligible w.r.t. the runtime of the SVM solvers. Note that training linear SVMs on the whitened kernel map is equivalent to training the kernel SVM classifier.

The experiments are performed on a laptop computer with an Intel CPU @ 1.8 GHz. Table 10 summarizes the results. The classification performance of SVM$^{\text{multi-class}}$ and OCAM is almost identical. The performance of both methods is slightly better than the results reported by Rieck et al. (2008), who used a heuristic one-against-all decomposition combined with the logistic regression. Comparison of the runtimes shows that the proposed OCAM is more than 10 times faster than the competing SVM$^{\text{multi-class}}$ and more than 20 times faster than the SVM$^{\text{light}}$ solver used in Rieck et al. (2008).

6. Conclusions

We have developed a novel method for solving large-scale risk minimization problems. Our proposed optimized cutting plane algorithm (OCA) extends the standard CPA algorithm of Teo et al. (2007) by, first, optimizing directly the primal problem via a line-search and, second, developing a new cutting plane selection strategy which significantly reduces the number of cutting planes needed to achieve an accurate solution. We have shown that the number of iterations OCA requires to converge to a ε-precise solution is approximately linear in the sample size. Applying OCA to two important learning problems, we obtained very fast specialized solvers for linear binary SVM classification (OCAS), and linear multi-class SVM classification (OCAM). In an extensive empirical evaluation on a large variety of problems comparing the proposed OCA with previous state-of-the-art SVM solvers, we achieved (depending on the task) speedups of up to three orders of magnitude obtaining the same precise SV solution. By parallelizing the subtasks of the algorithm, OCAS gained additional speedups of factors of up to 4.6 on a multi-core multiprocessor machine. Applying OCAS to a real-world splice site detection problem, we were able to train on a 12-million dimensional data set containing 50 million examples, achieving a new record performance for that task. Finally, we could reduce the training time on a malware classification problem by a factor of 20 over the previous approach. It remains as future work to derive OCAS for general structured output learning problems. Furthermore, we plan to extend OCAS to incorporate a bias term. Finally, it will be future work to investigate how the kernel framework can be incorporated into OCAS.

Acknowledgments

This work was supported in part by the FP7-ICT Programme of the European Union under the PASCAL2 Network of Excellence (ICT-216886) and by the Learning and Interference Platform of the Max Planck and Fraunhofer Societies. The main part of this work was done while VF was with the Fraunhofer IDA.FIRST when he was supported by a Marie Curie Intra-European Fellowship Grant SCOLES (MEIF-CT-2006-042107). VF was also supported by Czech Ministry of Education project 1M0567 during his current fellowship in the Center for Machine Perception. We thank Alexander Zien, Gunnar Rätsch, Konrad Rieck and Gilles Blanchard for great discussions. We also thank Konrad Rieck for providing us with the malware corpus.

Appendix A. Convergence Analysis

In this section we prove the convergence of OCA (Theorem 2). The core of the proof is adopted from Teo et al. (2007) who proved the Convergence Theorem 1 of the standard CPA. The main idea of the convergence theorem is based on deriving a lower bound on the improvement of the duality gap $F(\mathbf{w}_t^b) - F_t(\mathbf{w}_t) = \varepsilon_t$ and expressing this lower bound as a difference inequality (20), defined in Theorem 3. Having the difference inequality, (20) the proof of the convergence Theorem 2 follows easily.

The most laborious part is thus proving the auxiliary Theorem 3. The lower bound on the improvement $\varepsilon_t - \varepsilon_{t+1}$, which is the core inequality (20) in Theorem 3, is proven by showing that the objective value of the reduced problem solved at iteration $t+1$ must increase, provided the new added cutting plane violates the constraints of the reduced problem at iteration t. In the standard CPA, it is trivial to show that the new added cutting plane violates these constraints by at least ε_t.

Due to the sophisticated cutting plane selection strategy used in OCA, the violation of constraints is not obvious. Nevertheless, it can be proven, as we show in Lemma 1. Lemma 1 constitutes the main difference in the proofs of the standard CPA and the proposed OCA. The same lemma also explains why OCA converges faster than CPA. It will be shown that the minimal improvement of the reduced problem objective is a function of the constraint violation. While in the standard CPA the violation is guaranteed to be ε_t, the inequality (21) shows that the new cutting plane added in OCA violates the constraints by $\varepsilon_t + \frac{C}{2}\|\mathbf{w}_t^c - \mathbf{w}_t\|^2$. Unfortunately, we do not know how to bound the second term and thus the resulting lower bounds are the same for both algorithms.

The rest of this section is organized as follows. We first derive Lemmas 1, 2 and then we prove the auxiliary Theorem 3. Finally, we give the proof of the Convergence Theorem 2, which uses all previous results.

Theorem 3 *Assume that $\|\partial R(\mathbf{w})\| \leq G$ for all $\mathbf{w} \in \mathcal{W}$, where $\mathcal{W}$ is some domain of interest containing all $\mathbf{w}_{t'}$ for $t' \leq t$, and that $F(\mathbf{w}_t^b) - F_t(\mathbf{w}_t) = \varepsilon_t > 0$. In this case*

$$\varepsilon_t - \varepsilon_{t+1} \geq \frac{\varepsilon_t}{2} \min\left\{1, \frac{\varepsilon_t}{4C^2G^2}\right\}. \tag{20}$$

Lemma 1 *Let $F(\mathbf{w}_t^b) - F_t(\mathbf{w}_t) = \varepsilon_t > 0$. Then Algorithm 2 computes a new cutting plane $\langle \mathbf{w}, \mathbf{a}_{t+1}\rangle + b_{t+1} = 0$ which violates the constraints of the reduced problem (5) solved in the t-th iteration by at least $\frac{\varepsilon_t}{C}$, that is, it holds that*

$$C\big(b_{t+1} + \langle \mathbf{w}_t, \mathbf{a}_{t+1}\rangle - \xi_t\big) \geq \varepsilon_t + \frac{C}{2}\|\mathbf{w}_t^c - \mathbf{w}_t\|^2 \geq \varepsilon_t\,. \tag{21}$$

PROOF : We use the subgradient $\mathbf{w}_t^c + C\mathbf{a}_{t+1} \in \partial F(\mathbf{w}_t^c)$ to put a lower bound on the master objective F by means of a linear function at the point $\mathbf{w}_t^c$, that is,

$$f(\mathbf{w}) = F(\mathbf{w}_t^c) + \langle \mathbf{w}_t^c + C\mathbf{a}_{t+1}, \mathbf{w} - \mathbf{w}_t^c\rangle \leq F(\mathbf{w})\,, \qquad \forall \mathbf{w} \in \Re^n\,. \tag{22}$$

In Step 4 of Algorithm 2, the new best-so-far solution, $\mathbf{w}_t^b$, is computed as the minimizer of F over the line connecting the old best-so-far solution, $\mathbf{w}_{t-1}^b$, and the solution of reduced problem $\mathbf{w}_t$. In step 5, the new cutting plane $\langle \mathbf{w}, \mathbf{a}_{t+1}\rangle + b_{t+1} = 0$ is taken at the point $\mathbf{w}_t^c = \mathbf{w}_t^b(1-\mu) + \mathbf{w}_t\mu$, $\mu \in (0,1]$. Hence we conclude that $F(\mathbf{w}_t^b) \leq F(\mathbf{w}_t^c)$. Using the latter inequality and the lower bound (22), we obtain

$$f(\mathbf{w}_t^b) \leq F(\mathbf{w}_t^b) \leq F(\mathbf{w}_t^c) = f(\mathbf{w}_t^c)\,.$$

Since $\mathbf{w}_t^c$ lies on the line segment connecting $\mathbf{w}_t^b$ with $\mathbf{w}_t$ and because $f(\mathbf{w}_t^b) \leq f(\mathbf{w}_t^c)$ we conclude that

$$f(\mathbf{w}_t^c) \leq f(\mathbf{w}_t)\,. \tag{23}$$

Note that the inequality (23) holds only for $\mu \in (0,1]$. Using (22) we can rewrite (23) as

$$F(\mathbf{w}_t^c) \leq F(\mathbf{w}_t^c) + \langle \mathbf{w}_t^c + C\mathbf{a}_{t+1}, \mathbf{w}_t - \mathbf{w}_t^c\rangle\,. \tag{24}$$

Combining $F(\mathbf{w}_t^b) \leq F(\mathbf{w}_t^c)$ and $F(\mathbf{w}_t^b) - F_t(\mathbf{w}_t) = \varepsilon_t$ we get $F(\mathbf{w}_t^c) - F_t(\mathbf{w}_t) \geq \varepsilon_t$. Finally, substituting (24) to the latter inequality yields

$$F(\mathbf{w}_t^c) + \langle \mathbf{w}_t^c + C\mathbf{a}_{t+1}, \mathbf{w}_t - \mathbf{w}_t^c\rangle - F_t(\mathbf{w}_t) \geq \varepsilon_t\,,$$

which can be further rearranged into (21). ■

Lemma 2 *(Teo et al., 2007) Let $\Delta(\tau) = l\tau - \frac{h}{2}\tau^2$ be a concave quadratic function such that $\Delta'(0) \geq L > 0$ and $|\Delta''(\tau)| = h \leq H$, $\forall \tau \in [0,1]$. Then the maximal value of Δ attained for the interval $[0,1]$ has a lower bound defined by*

$$\max_{\tau \in [0,1]} \Delta(\tau) \geq \frac{L}{2} \min\left(1, \frac{L}{H}\right).$$

PROOF : Using $l \geq L$ and $h \leq H$ we obtain a lower bound $\Delta(\tau)$ by $L\tau - \frac{H}{2}\tau^2$. The unconstrained maximum of the lower bound is attained at point $\frac{L}{H}$, which leads to the value of $\frac{L^2}{2H}$. If $\frac{L}{H} > 1$ then the constrained maximum of the lower bound is attained at 1, which yields the maximal value of $L - \frac{H}{2}$. Using $\frac{L}{H} > 1$, the value of $L - \frac{H}{2}$ has a lower bound of $\frac{L}{2}$. Taking the minimum over both maxima proves the claim. ■

PROOF OF THEOREM 3: We can put a lower bound on the difference $\varepsilon_t - \varepsilon_{t+1}$ using the improvement of the dual objective function $D_{t+1}(\alpha_{t+1}) - D_t(\alpha_t)$ because

$$\begin{aligned} \varepsilon_t - \varepsilon_{t+1} &= F(\mathbf{w}_t^b) - F_t(\mathbf{w}_t) - F(\mathbf{w}_{t+1}^b) + F_{t+1}(\mathbf{w}_{t+1}) \\ &\geq F_{t+1}(\mathbf{w}_{t+1}) - F_t(\mathbf{w}_t) \\ &= D_{t+1}(\alpha_{t+1}) - D_t(\alpha_t). \end{aligned}$$

The inequality follows after excluding the term $F(\mathbf{w}_t^b) - F(\mathbf{w}_{t+1}^b) \geq 0$ and the last equality is the result of the fact that the primal and dual optimal values are equal.

The value of $D_{t+1}(\alpha_{t+1})$ is defined as the maximum of D_{t+1} w.r.t. the convex feasible set $\mathcal{A}_{t+1}$. Hence, by maximizing D_{t+1} w.r.t. a line segment lying entirely inside $\mathcal{A}_{t+1}$ we get a lower bound on $D_{t+1}(\alpha_{t+1})$, that is,

$$D_{t+1}(\alpha_{t+1}) - D_t(\alpha_t) \geq \max_{\tau \in [0,1]} \Delta(\tau) := D_{t+1}\big(\beta(1-\tau) + \gamma\tau\big) - D_t(\alpha_t),$$

where β and γ are arbitrary vectors from $\mathcal{A}_{t+1}$. Specifically, we define the vectors as

$$\beta = (\alpha_t; 0) \in \Re^{t+1} \quad \text{and} \quad \gamma = (\mathbf{0}; C) \in \Re^{t+1}. \tag{25}$$

Now we show that $\Delta(\tau)$ is a function compliant with the assumptions of Lemma 2, which will allow us to lower bound its value for the interval $[0,1]$. To this end, we need to derive the explicit form of $\Delta(\tau)$ and then compute $\Delta'(0)$ and an upper bound on $\Delta''(\tau)$, $\forall \tau \in [0,1]$. Defining a vector $\mathbf{b} = (b_1; \ldots; b_{t+1}) \in \Re^{t+1}$ and a matrix $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_{t+1}) \in \Re^{n \times (t+1)}$ we can write the objective of the dual of the reduced problem as $D_{t+1}(\alpha) = \langle \alpha, \mathbf{b} \rangle - \frac{1}{2}\|\mathbf{A}\alpha\|^2$. Using the latter definition of D_{t+1} and (25), we can rewrite $\Delta(\tau)$ as

$$\Delta(\tau) = \tau\big\langle \gamma - \beta, \mathbf{b} - \mathbf{A}^T\mathbf{A}\beta \big\rangle - \frac{1}{2}\tau^2 \big\|\mathbf{A}\beta - \mathbf{A}\gamma\big\|^2.$$

The value of the derivative $\Delta'(0)$ can be written as

$$\Delta'(0) = \big\langle \gamma - \beta, \mathbf{b} - \mathbf{A}^T\mathbf{A}\beta \big\rangle = C\big(b_{t+1} + \langle \mathbf{w}_t, \mathbf{a}_{t+1} \rangle - \xi_t\big). \tag{26}$$

The second equality of (26) was derived by using (25), $\mathbf{w}_t = -\sum_{i=1}^t \mathbf{a}_i[\alpha_t]_i$ and $F_t(\mathbf{w}_t) = D_t(\alpha_t)$. Using Lemma 1, we get a lower bound of the right-hand side of (26), that is

$$\Delta'(0) \geq \varepsilon_t. \tag{27}$$

The absolute value of the second derivative $|\Delta''(\tau)|$ can be upper bound by

$$|\Delta''(\tau)| = \|\mathbf{A}\beta - \mathbf{A}\gamma\|^2 \leq \|\mathbf{A}\beta\|^2 + \|\mathbf{A}\gamma\|^2 \leq 4C^2 \max_{i=1,\ldots,t+1} \|\mathbf{a}_i\|^2 \leq 4C^2G^2\,, \tag{28}$$

where we use the assumption $\mathbf{a}_i = \|\partial F(\mathbf{w})\| \leq G$ and the fact that the vector $\frac{1}{C}\mathbf{A}\alpha$ equals the convex combination of the columns of $\mathbf{A}$ for any $\alpha \in \mathcal{A}_{t+1}$, hence, its norm cannot be greater than $\max_i \|\mathbf{a}_i\|$. Finally, using (27) and (28) in Lemma 2 yields the claim of Theorem 3. ■

PROOF OF THEOREM 2: The proof is adopted from Teo et al. (2007). For any $\varepsilon_t > 4C^2G^2$ it follows from (20) that $\varepsilon_{t+1} \leq \frac{\varepsilon_t}{2}$. Moreover, $\varepsilon_0 \leq F(\mathbf{0})$, since F is nonnegative. Hence, we need at most $\log_2 \frac{F(\mathbf{0})}{4C^2G^2}$ iterations to achieve a level of precision better than $4C^2G^2$. Subsequently, we need to solve the following difference equation:

$$\varepsilon_{t+1} - \varepsilon_t = -\frac{\varepsilon_t^2}{8C^2G^2}\,.$$

Since this is monotonically decreasing, we can upper bound this by solving the differential equation $\varepsilon'(t) = -\frac{\varepsilon^2(t)}{8C^2G^2}$, with the boundary condition $\varepsilon(0) = 4C^2G^2$. This in turn yields $\varepsilon(t) = \frac{8C^2G^2}{t+2}$, and hence $t \leq \frac{8C^2G^2}{\varepsilon} - 2$ to achieve $\varepsilon(t) \leq \varepsilon$. For a given ε we will need $\frac{8C^2G^2}{\varepsilon} - 2$ more iterations to converge. This proves the claim.

■

References

A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with LaRank. In *Proceedings of International Machine Learning Conference (ICML)*, pages 89 – 96. OmniPress, 2007.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 161 – 168. MIT Press, 2007.

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`.

O. Chapelle. Training a Support Vector Machine in the Primal. *Neural Computation*, 19(5):1155–1178, 2007.

M. Collins, R.E. Schapire, and Y. Singer. Logistic regression AdaBoost and Bregman distance. In *Proceedings of Annual Conference on Computational Learning Theory (COLT)*, pages 158–169. Morgan Kaufman, San Francisco, 2000.

C. Cortes and V.N. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge UP, Cambridge, UK, 2000.

S. Degroeve, Y. Saeys, P. De Baets, B. Rouzé, and Y. Van de Peer. SpliceMachine: predicting splice sites from high-dimensional local context representations. *Bioinformatics*, 21(8):1332–8, 2005.

R. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

R.E. Fan, P.H. Chen, and C.J. Lin. Working set selection using second order information for training SVM. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

T. Fawcett. ROC graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Laboratories, Palo Alto, CA, USA, January 2003.

V. Franc. *Optimization Algorithms for Kernel Methods*. PhD thesis, Czech Technical University in Prague, July 2005. Supervised by V. Hlaváč.

V. Franc and S. Sonnenburg. OCAS optimized cutting plane algorithm for support vector machines. In *Proceedings of International Machine Learning Conference (ICML)*, pages 320–327. ACM Press, 2008a.

V. Franc and S. Sonnenburg. LIBOCAS, 2008b. Software available at `http://mloss.org/software/view/85/`.

V. Franc, P. Laskov, and K.-R. Müller. Stopping conditions for exact computation of leave-one-out error in support vector machines. In *Proceedings of International Machine Learning Conference (ICML)*, pages 328–335. ACM Press, 2008.

T. Joachims. Making large–scale SVM learning practical. In *Advances in Kernel Methods — Support Vector Learning*, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 377 – 384. ACM New York, NY, USA, 2005.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 217 – 226. ACM New York, NY, USA, 2006.

T. Joachims, T. Finley, and C.N. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 76(1), May 2009.

C. Leslie, E. Eskin, and W.S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of Pacific Biocomputing Symposium (PBS)*, pages 564–575. River Edge, NJ, World Scientific, 2002.

C.J. Lin, R.C. Weng, and S.S. Keerthi. Trust region Newton methods for large-scale logistic regression. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 561 – 568. ACM Press New York, 2007.

G. Rätsch and S. Sonnenburg. Accurate splice site detection for *Caenorhabditis elegans*. In K. Tsuda B. Schölkopf and J.-P. Vert, editors, *Kernel Methods in Computational Biology*. MIT Press, 2004.

G. Rätsch, S. S. Sonnenburg, and B. Schölkopf. RASE: recognition of alternatively spliced exons in *C. elegans*. *Bioinformatics*, 21(Suppl. 1):i369–i377, June 2005.

K. Rieck, T. Holtz, C. Willems, P. Düssel, and P. Laskov. Learning and classification of malware behaviour. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), Fifth International Conference*, pages 108–125, July 2008.

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

B. Schölkopf, J. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson. Estimating the support of a high-dimensional distribution. Technical Report TR 87, Microsoft Research, Redmond, WA, 1999.

S.S. Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 807 – 814. ACM Press, 2007.

V. Sindhwani and S.S. Keerthi. Newton methods for fast solution of semi-supervised linear svms. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*. MIT Press, 2007.

S. Sonnenburg. New methods for splice site recognition. Master's thesis, Humboldt University, 2002. supervised by K.-R. Müller H.-D. Burkhard and G.Rätsch.

S. Sonnenburg. *Machine Learning for Genomic Sequence Analysis*. PhD thesis, Fraunhofer Institute FIRST, 2008. supervised by K.-R. Müller and G.Rätsch.

S. Sonnenburg and G. Rätsch. Shogun, 2007. Software available at `http://mloss.org/software/view/2/`.

S. Sonnenburg, G. Rätsch, and K. Rieck. Large scale learning with string kernels. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*. MIT Press, 2007a.

S. Sonnenburg, G. Schweikert, P. Philips, J. Behr, and G. Rätsch. Accurate Splice Site Prediction. *BMC Bioinformatics, Special Issue from NIPS workshop on New Problems and Methods in Computational Biology Whistler, Canada, 18 December 2006*, 8:(Suppl. 10):S7, December 2007b.

S. Sonnenburg, V. Franc, E. Yomtov, and M. Sebag. The pascal large scale learning challenge. *Journal of Machine Learning Research*, 2009. (manuscript in preparation).

C.H. Teo, Q. Le, A. Smola, and S.V.N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD)*, August 2007.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, Sep. 2005.

Wikipedia. DDR2 SDRAM — Wikipedia, the free encyclopedia, 2009. URL `{http://en.wikipedia.org/wiki/DDR2_SDRAM}`. [Online; accessed 5-February-2009].

C.K.I Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In *Learning and Inference in Graphical Models*, pages 599–621. Kluwer Academic, 1998.

L. Zanni, T. Serafini, and G. Zanghirati. Parallel software for training. *Journal of Machine Learning Research*, 7:1467–1492, July 2006.

Journal of Machine Learning Research 10 (2009) 2193-2232 Submitted 2/08; Revised 7/09; Published 10/09

Margin-based Ranking and an Equivalence between AdaBoost and RankBoost

Cynthia Rudin* RUDIN@MIT.EDU
MIT Sloan School of Management
Cambridge, MA 02142

Robert E. Schapire SCHAPIRE@CS.PRINCETON.EDU
Department of Computer Science
35 Olden Street
Princeton University
Princeton NJ 08540

Editor: Nicolas Vayatis

Abstract

We study boosting algorithms for learning to rank. We give a general margin-based bound for ranking based on covering numbers for the hypothesis space. Our bound suggests that algorithms that maximize the ranking margin will generalize well. We then describe a new algorithm, smooth margin ranking, that precisely converges to a maximum ranking-margin solution. The algorithm is a modification of RankBoost, analogous to "approximate coordinate ascent boosting." Finally, we prove that AdaBoost and RankBoost are equally good for the problems of bipartite ranking and classification in terms of their asymptotic behavior on the training set. Under natural conditions, AdaBoost achieves an area under the ROC curve that is equally as good as RankBoost's; furthermore, RankBoost, when given a specific intercept, achieves a misclassification error that is as good as AdaBoost's. This may help to explain the empirical observations made by Cortes and Mohri, and Caruana and Niculescu-Mizil, about the excellent performance of AdaBoost as a bipartite ranking algorithm, as measured by the area under the ROC curve.

Keywords: ranking, RankBoost, generalization bounds, AdaBoost, area under the ROC curve

1. Introduction

Consider the following supervised learning problem: Sylvia would like to get some recommendations for good movies before she goes to the theater. She would like a ranked list that agrees with her tastes as closely as possible, since she will probably go to see the movie closest to the top of the list that is playing at the local theater. She does not want to waste her time and money on a movie she probably will not like.

The information she provides is as follows: for many different pairs of movies she has seen, she will tell the learning algorithm whether or not she likes the first movie better than the second one.[1] This allows her to rank whichever pairs of movies she wishes, allowing for the possibility of ties

*. Also at Center for Computational Learning Systems, Columbia University, 475 Riverside Drive MC 7717, New York, NY 10115.

1. In practice, she could simply rate the movies, but this gives pairwise information also. The pairwise setting is strictly more general in this sense.

between movies, and the possibility that certain movies cannot necessarily be compared by her (for instance, she may not wish to directly compare cartoons with action movies). Sylvia does not need to be consistent, in the sense that she may rank $\mathbf{a} > \mathbf{b} > \mathbf{c} > \mathbf{a}$. (The loss function and algorithm will accommodate this. See Martin Gardner's amusing article (Gardner, 2001) on how nontransitivity can arise naturally in many situations.) Each pair of movies such that Sylvia ranks the first above the second is called a "crucial pair."

The learning algorithm has access to a set of n individuals, called "weak rankers" or "ranking features," who have also ranked pairs of movies. The learning algorithm must try to combine the views of the weak rankers in order to match Sylvia's preferences, and generate a recommendation list that will generalize her views. In this paper, our goal is to design and study learning algorithms for ranking problems such as this collaborative filtering task.

The ranking problem was studied in depth by Freund et al. (2003), where the RankBoost algorithm was introduced. In this setting, the ranked list is constructed using a linear combination of the weak rankers. Ideally, this combination should minimize the probability that a crucial pair is misranked, that is, the probability that the second movie in the crucial pair is ranked above the first. RankBoost aims to minimize an exponentiated version of this misranking probability.

A special case of the general ranking problem is the "bipartite" ranking problem, where there are only two classes: a positive class (good movies) and a negative class (bad movies). In this case, the misranking probability is the probability that a good movie will be ranked below a bad movie. This quantity is an affine transformation of the (negative of the) area under the Receiver-Operator-Characteristic curve (AUC).

Bipartite ranking is different from the problem of classification; if, for a given data set, the misclassification error is zero, then the misranking error must also be zero, but the converse is not necessarily true. For the ranking problem, the examples are viewed relative to each other and the decision boundary is irrelevant.

Having described the learning setting, we can now briefly summarize our three main results.

- *Generalization bound:* In Section 3, we provide a margin-based bound for ranking in the general setting described above. Our ranking margin is defined in analogy with the classification margin, and the complexity measure for the hypothesis space is a "sloppy covering number," which yields, as a corollary, a bound in terms of the L_∞ covering number. Our bound indicates that algorithms that maximize the margin will generalize well.

- *Smooth margin ranking algorithm:* We present a ranking algorithm in Section 4 designed to maximize the margin. Our algorithm is based on a "smooth margin," and we present an analysis of its convergence.

- *An equivalence between AdaBoost and RankBoost:* A remarkable property of AdaBoost is that it not only solves the classification problem, but simultaneously solves the same problem of bipartite ranking as its counterpart, RankBoost. This is proved in Section 5. One does not need to alter AdaBoost in any way for this property to hold. Conversely, the solution of RankBoost can be slightly altered to achieve a misclassification loss that is equally as good as AdaBoost's.

We now provide some background and related results.

Generalization bounds are useful for showing that an algorithm can generalize beyond its training set, in other words, that prediction is possible. More specifically, bounds indicate that a small

probability of error will most likely be achieved through a proper balance of the empirical error and the complexity of the hypothesis space. This complexity can by measured by many informative quantities; for instance, the VC dimension, which is linked in a fundamental way to classification, and the Rademacher and Gaussian complexities (Bartlett and Mendelson, 2002). The use of these quantities is tied to a kind of natural symmetry that typically exists in such problems, for instance, in the way that positive and negative examples are treated symmetrically in a classification setting. The limited bipartite case has this symmetry, but not the more general ranking problem that we have described. Prior bounds on ranking have either made approximations in order to use the VC Dimension for the general problem (as discussed by Clemençon et al., 2005, 2007, who work on statistical aspects of ranking) or focused on the bipartite case (Freund et al., 2003; Agarwal et al., 2005; Usunier et al., 2005). For our bound, we choose a covering number in the spirit of Bartlett (1998). The covering number is a general measure of the capacity of the hypothesis space; it does not lend itself naturally to classification like the VC dimension, is not limited to bipartite ranking, nor does it require symmetry in the problem. Thus, we are able to work around the lack of symmetry in this setting. In fact, a preliminary version of our work (Rudin et al., 2005) has been extended to a highly nonsymmetric setting, namely the case where the top part of the list is considered more important (Rudin, 2009). Several other recent works also consider this type of highly nonsymmetric setting for ranking (Dekel et al., 2004; Cossock and Zhang, 2008; Clemençon and Vayatis, 2007; Shalev-Shwartz and Singer, 2006; Le and Smola, 2007).

When deriving generalization bounds, it is important to consider the "separable" case, where all training instances are correctly handled by the learning algorithm so that the empirical error is zero. In the case of bipartite ranking, the separable case means that all positive instances are ranked above all negative instances, and the area under the ROC curve is exactly 1. In the separable case for classification, one important indicator of a classifier's generalization ability is the "margin." The margin has proven to be an important quantity in practice for determining an algorithm's generalization ability, for example, in the case of AdaBoost (Freund and Schapire, 1997) and support vector machines (SVMs) (Cortes and Vapnik, 1995). Although there has been some work devoted to generalization bounds for ranking as we have mentioned (Clemençon et al., 2005, 2007; Freund et al., 2003; Agarwal et al., 2005; Usunier et al., 2005), the bounds that we are aware of are not margin-based, and thus do not provide this useful type of discrimination between ranking algorithms in the separable case.

Since we are providing a general margin-based bound for ranking in Section 3, we derive algorithms which create large margins. For the classification problem, it was proved that AdaBoost does not always fully maximize the (classification) margin (Rudin et al., 2004). In fact, AdaBoost does not even necessarily make progress towards increasing the margin at every iteration. Since AdaBoost (for the classification setting) and RankBoost (for the ranking setting) were derived analogously for the two settings, RankBoost does not directly maximize the ranking margin, and it does not necessarily increase the margin at every iteration. In Section 4.1 we introduce a "smooth margin" ranking algorithm, and prove that it makes progress towards increasing the smooth margin for ranking at every iteration; this is the main step needed in proving convergence and convergence rates. This algorithm is analogous to the smooth margin classification algorithm "approximate coordinate ascent boosting" (Rudin et al., 2007) in its derivation, but the analogous proof that progress occurs at each iteration is much trickier; hence we present this proof here, along with a theorem stating that this algorithm converges to a maximum margin solution.

Although AdaBoost and RankBoost were derived analogously for the two settings, the parallels between AdaBoost and RankBoost are deeper than their derivations. A number of papers, including those of Cortes and Mohri (2004) and Caruana and Niculescu-Mizil (2006) have noted that in fact, AdaBoost experimentally seems to be very good at the bipartite ranking problem, even though it was RankBoost that was explicitly designed to solve this problem, not AdaBoost. Or, stated another way, AdaBoost often achieves a large area under the ROC curve. In Section 5, we present a possible explanation for these experimental observations. Namely, we show that if the weak learning algorithm is capable of producing the constant classifier (the classifier whose value is always one), then remarkably, AdaBoost and RankBoost produce equally good solutions to the ranking problem in terms of loss minimization and area under the ROC curve on the training set. More generally, we define a quantity called "F-skew," an exponentiated version of the "skew" used in the expressions of Cortes and Mohri (2004, 2005) and Agarwal et al. (2005). If the F-skew vanishes, AdaBoost minimizes the exponentiated ranking loss, which is the same loss that RankBoost explicitly minimizes; thus, the two algorithms will produce equally good solutions to the exponentiated problem. Moreover, if AdaBoost's set of weak classifiers includes the constant classifier, the F-skew always vanishes. From there, it is only a small calculation to show that AdaBoost and RankBoost achieve the same asymptotic AUC value whenever it can be defined. An analogous result does not seem to hold true for support vector machines; SVMs designed to maximize the AUC only seem to yield the same AUC as the "vanilla" classification SVM in the separable case, when the AUC is exactly one (Rakotomamonjy, 2004; Brefeld and Scheffer, 2005). The main result may be useful for practitioners: if the cost of using RankBoost is prohibitive, it may be useful to consider AdaBoost to solve the ranking problem.

The converse result also holds, namely that a solution of RankBoost can be slightly modified so that the F-skew vanishes, and the asymptotic misclassification loss is equal to AdaBoost's whenever it can be defined.

We proceed from the most general to the most specific. First, in Section 3 we provide a margin-based bound for general ranking. In Sections 4.1 and 4.2 we fix the form of the hypothesis space to match that of RankBoost, that is, the space of binary functions. Here, we discuss RankBoost, AdaBoost and other coordinate-based ranking algorithms, and introduce the smooth margin ranking algorithm. In Section 5, we focus on the bipartite ranking problem, and discuss conditions for AdaBoost to act as a bipartite ranking algorithm by minimizing the exponentiated loss associated with the AUC. Sections 3 and 4.2 focus on the separable case where the training error vanishes, and Sections 4.1 and 5 focus on the nonseparable case. Sections 6, 7, and 8 contain the major proofs.

A preliminary version of this work appeared in a conference paper with Cortes and Mohri (Rudin et al., 2005). Many of the results from that work have been made more general here.

2. Notation

We use notation similar to Freund et al. (2003). The training data for the supervised ranking problem consists of *instances* and their *truth function* values. The *instances*, denoted by S, are $\{\mathbf{x}_i\}_{i=1,\ldots,m}$, where $\mathbf{x}_i \in \mathcal{X}$ for all i. The set $\mathcal{X}$ is arbitrary and may be finite or infinite, usually $\mathcal{X} \subset \mathbb{R}^N$. In the case of the movie ranking problem, the $\mathbf{x}_i$'s are the movies and $\mathcal{X}$ is the set of all possible movies. We assume $\mathbf{x}_i \in \mathcal{X}$ are chosen independently and at random (iid) from a fixed but unknown probability distribution $\mathcal{D}$ on $\mathcal{X}$ (assuming implicitly that anything that needs to be measurable is measurable). The notation $\mathbf{x} \sim \mathcal{D}$ means $\mathbf{x}$ is chosen randomly according to distribution $\mathcal{D}$. The notation $S \sim \mathcal{D}^m$

means each of the m elements of the training set S are chosen independently at random according to $\mathcal{D}$.

The values of the *truth function* $\pi : \mathcal{X} \times \mathcal{X} \to \{0,1\}$, which is defined over pairs of instances, are analogous to the "labels" in classification. If $\pi(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = 1$, this means that the pair $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$ is a *crucial pair*: $\mathbf{x}^{(1)}$ should be ranked more highly than $\mathbf{x}^{(2)}$. We will consider a non-noisy case where π is deterministic, which means $\pi(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) = 0$, meaning that $\mathbf{x}^{(1)}$ should not be ranked higher than itself, and also that $\pi(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = 1$ implies $\pi(\mathbf{x}^{(2)}, \mathbf{x}^{(1)}) = 0$, meaning that if $\mathbf{x}^{(1)}$ is ranked more highly than $\mathbf{x}^{(2)}$, then $\mathbf{x}^{(2)}$ should not be ranked more highly than $\mathbf{x}^{(1)}$. It is possible to have $\pi(\mathbf{a}, \mathbf{b}) = 1$, $\pi(\mathbf{b}, \mathbf{c}) = 1$, and $\pi(\mathbf{c}, \mathbf{a}) = 1$, in which case the algorithm will always suffer some loss; we will be in the nonseparable case when this occurs. The total number of crucial training pairs can be no larger than $m(m-1)/2$ based on the rules of π, and should intuitively be of the order m^2 in order for us to perform ranking with sufficient accuracy. We assume that for each pair of training instances $\mathbf{x}_i, \mathbf{x}_k$ we receive, we also receive the value of $\pi(\mathbf{x}_i, \mathbf{x}_k)$. In a more general model, we allow the value $\pi(\mathbf{x}_i, \mathbf{x}_k)$ to be generated probabilisitically conditional on each training pair $\mathbf{x}_i, \mathbf{x}_k$. For the generalization bounds in this paper, for simplicity of presentation, we do not consider this more general model, although all of our results can be shown to hold in the more general case as well. The quantity $E := \mathbb{E}_{\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \sim \mathcal{D}}[\pi(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})]$ is the expected proportion of pairs in the database that are crucial pairs, $0 \leq E \leq 1/2$.

Back to the collaborative filtering example, to obtain the training set, Sylvia is given a random sample of movies, chosen randomly from the distribution of movies being shown in the theater. Sylvia must see these training movies and tell us all pairs of these movies such that she would rank the first above the second to determine values of the truth function π.

Our goal is to construct a ranking function $f : \mathcal{X} \to \mathbb{R}$, which gives a real valued score to each instance in $\mathcal{X}$. We do not care about the actual values of each instance, only the relative values; for instance, we do not care if $f(\mathbf{x}^{(1)}) = .4$ and $f(\mathbf{x}^{(2)}) = .1$, only that $f(\mathbf{x}^{(1)}) > f(\mathbf{x}^{(2)})$, which we interpret to mean that $\mathbf{x}^{(1)}$ is predicted by f to be ranked higher (better) than $\mathbf{x}^{(2)}$. Also, the function f should be bounded, $f \in L_\infty(\mathcal{X})$ (or in the case where $|\mathcal{X}|$ is finite, $f \in \ell_\infty(\mathcal{X})$).

In the usual setting of boosting for classification, $|f(\mathbf{x})| \leq 1$ for all $\mathbf{x}$ and the *margin of training instance i* (with respect to classifier f) is defined by Schapire et al. (1998) to be $y_i f(\mathbf{x}_i)$, where y_i is the classification label, $y_i \in \{-1, 1\}$. The *margin of classifier f* is defined to be the minimum margin over all training instances, $\min_i y_i f(\mathbf{x}_i)$. Intuitively, the margin tells us how much the classifier f can change before one of the training instances is misclassified; it gives us a notion of how stable the classifier is.

For the ranking setting, we define an analogous notion of margin. Here, we normalize our bounded function f so that $0 \leq f \leq 1$. The *margin of crucial pair* $\mathbf{x}_i$,$\mathbf{x}_k$ (with respect to ranking function f) will be defined as $f(\mathbf{x}_i) - f(\mathbf{x}_k)$. The *margin of ranking function f*, is defined to be the minimum margin over all crucial pairs,

$$\text{margin}_f := \mu_f := \min_{\{i,k|\pi(\mathbf{x}_i,\mathbf{x}_k)=1\}} f(\mathbf{x}_i) - f(\mathbf{x}_k).$$

Intuitively, the margin tells us how much the ranking function can change before one of the crucial pairs is misranked. As with classification, we are in the separable case whenever the margin of f is positive.

In Section 5 we will discuss the problem of bipartite ranking. Bipartite ranking is a subset of the general ranking framework we have introduced. In the bipartite ranking problem, every training

instance falls into one of two categories, the positive class Y_+ and the negative class Y_-. To transform this into the general framework, take $\pi(\mathbf{x}_i, \mathbf{x}_k) = 1$ for each pair $i \in Y_+$ and $k \in Y_-$. That is, a crucial pair exists between an element of the positive class and an element of the negative class. The class of each instance is assumed deterministic, consistent with the setup described earlier. Again, the results can be shown to hold in the case of nondeterministic class labels.

It may be tempting to think of the ranking framework as if it were just classification over the space $\mathcal{X} \times \mathcal{X}$. However, this is not the case; the examples are assumed to be drawn randomly from $\mathcal{X}$, rather than pairs of examples drawn from $\mathcal{X} \times \mathcal{X}$. Furthermore, the scoring function f has domain $\mathcal{X}$, that is, in order to produce a single ranked list, we should have $f : \mathcal{X} \to \mathbb{R}$ rather than $f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. In the latter case, one would need an additional mechanism to reconcile the scores to produce a single ranked list. Furthermore, the bipartite ranking problem does not have the same goal as classification even though the labels are $\{-1, +1\}$. In classification, the important quantity is the misclassification error involving the sign of f, whereas for bipartite ranking, the important quantity is perhaps the area under the ROC curve, relying on differences between f values. A change in the position of one example can change the bipartite ranking loss without changing the misclassification error and vice versa.

3. A Margin-Based Bound for Ranking

Bounds in learning theory are useful for telling us which quantities (such as the margin) are involved in the learning process (see Bousquet, 2003, for discussion on this matter). In this section, we provide a margin-based bound for ranking, which gives us an intuition for separable-case ranking and yields theoretical encouragement for margin-based ranking algorithms. The quantity we hope to minimize here is the misranking probability; for two randomly chosen instances, if they are a crucial pair, we want to minimize the probability that these instances will be misranked. Formally, this misranking probability is:

$$\begin{aligned}\mathbb{P}_{\mathcal{D}}\{\text{misrank}_f\} &:= \mathbb{P}_{\mathcal{D}}\{f(\bar{\mathbf{x}}) \le f(\tilde{\mathbf{x}}) \mid \pi(\bar{\mathbf{x}}, \tilde{\mathbf{x}}) = 1\} = \mathbb{E}_{\bar{\mathbf{x}}, \tilde{\mathbf{x}} \sim \mathcal{D}}[\mathbf{1}_{[f(\bar{\mathbf{x}}) \le f(\tilde{\mathbf{x}})]} \mid \pi(\bar{\mathbf{x}}, \tilde{\mathbf{x}}) = 1] \\ &= \frac{\mathbb{E}_{\bar{\mathbf{x}}, \tilde{\mathbf{x}} \sim \mathcal{D}}[\mathbf{1}_{[f(\bar{\mathbf{x}}) \le f(\tilde{\mathbf{x}})]} \pi(\bar{\mathbf{x}}, \tilde{\mathbf{x}})]}{\mathbb{E}_{\bar{\mathbf{x}}, \tilde{\mathbf{x}} \sim \mathcal{D}}[\pi(\bar{\mathbf{x}}, \tilde{\mathbf{x}})]} = \frac{\mathbb{E}_{\bar{\mathbf{x}}, \tilde{\mathbf{x}} \sim \mathcal{D}}[\mathbf{1}_{[f(\bar{\mathbf{x}}) \le f(\tilde{\mathbf{x}})]} \pi(\bar{\mathbf{x}}, \tilde{\mathbf{x}})]}{E}. \end{aligned} \tag{1}$$

The numerator of (1) is the fraction of pairs that are both crucial and incorrectly ranked by f, and the denominator, $E := \mathbb{E}_{\bar{\mathbf{x}}, \tilde{\mathbf{x}} \sim \mathcal{D}}[\pi(\bar{\mathbf{x}}, \tilde{\mathbf{x}})]$ is the fraction of pairs that are crucial pairs. Thus, $\mathbb{P}_{\mathcal{D}}\{\text{misrank}_f\}$ is the fraction of crucial pairs that are incorrectly ranked by f.

Since we do not know $\mathcal{D}$, we may calculate only empirical quantities that rely only on our training sample. An empirical quantity that is analogous to $\mathbb{P}_{\mathcal{D}}\{\text{misrank}_f\}$ is the following:

$$\begin{aligned}\mathbb{P}_S\{\text{misrank}_f\} &:= \mathbb{P}_S\{\text{margin}_f \le 0\} := \mathbb{P}_S\{f(\mathbf{x}_i) \le f(\mathbf{x}_k) \mid \pi(\mathbf{x}_i, \mathbf{x}_k) = 1\} \\ &= \frac{\sum_{i=1}^m \sum_{k=1}^m \mathbf{1}_{[f(\mathbf{x}_i) \le f(\mathbf{x}_k)]} \pi(\mathbf{x}_i, \mathbf{x}_k)}{\sum_{i=1}^m \sum_{k=1}^m \pi(\mathbf{x}_i, \mathbf{x}_k)}. \end{aligned}$$

We make this terminology more general, by allowing it to include a margin of θ. For the bound we take $\theta > 0$:

$$\begin{aligned}\mathbb{P}_S\{\text{margin}_f \le \theta\} &:= \mathbb{P}_S\{f(\mathbf{x}_i) - f(\mathbf{x}_k) \le \theta \mid \pi(\mathbf{x}_i, \mathbf{x}_k) = 1\} \\ &= \frac{\sum_{i=1}^m \sum_{k=1}^m \mathbf{1}_{[f(\mathbf{x}_i) - f(\mathbf{x}_k) \le \theta]} \pi(\mathbf{x}_i, \mathbf{x}_k)}{\sum_{i=1}^m \sum_{k=1}^m \pi(\mathbf{x}_i, \mathbf{x}_k)}, \end{aligned}$$

that is, $\mathbb{P}_S\{\text{margin}_f \leq \theta\}$ is the fraction of crucial pairs in $S \times S$ with margin not larger than θ.

We want to bound $\mathbb{P}_{\mathcal{D}}\{\text{misrank}_f\}$ in terms of an empirical, margin-based term and a complexity term. The type of complexity we choose is a "sloppy covering number" of the sort used by Schapire et al. (1998). Since such a covering number can be bounded by an L_∞ covering number, we will immediately obtain L_∞ covering number bounds as well, including a strict improvement on the one derived in the preliminary version of our work (Rudin et al., 2005). Here, we implicitly assume that $\mathcal{F} \subset L_\infty(\mathcal{X})$, $f \in \mathcal{F}$ are everywhere defined.

We next define sloppy covers and sloppy covering numbers.

Definition 1 *For $\varepsilon, \theta \geq 0$, a set $\mathcal{G}$ is a* θ-sloppy ε-cover *for $\mathcal{F}$ if for all $f \in \mathcal{F}$ and for all probability distributions $\mathcal{D}$ on $\mathcal{X}$, there exists $g \in \mathcal{G}$ such that*

$$\mathbb{P}_{\mathbf{x}\sim D}[|f(\mathbf{x}) - g(\mathbf{x})| \geq \theta] \leq \varepsilon.$$

The corresponding sloppy covering number *is the size of the smallest θ-sloppy ε-cover $\mathcal{G}$, and is written $\mathcal{N}(\mathcal{F}, \theta, \varepsilon)$.*

The *L_∞ covering number* $\mathcal{N}_\infty(\mathcal{F}, \varepsilon)$ is defined as the minimum number of (open) balls of radius ε needed to cover $\mathcal{F}$, using the L_∞ metric. Since $\|f - g\|_\infty < \theta$ implies that $\mathbb{P}_{\mathbf{x}\sim D}[|f(\mathbf{x}) - g(\mathbf{x})| \geq \theta] = 0$, we have that the sloppy covering number $\mathcal{N}(\mathcal{F}, \theta, \varepsilon)$ is never more than $\mathcal{N}_\infty(\mathcal{F}, \theta)$, and in some cases it can be exponentially smaller, such as for convex combinations of binary functions as discussed below.

Here is our main theorem, which is proved in Section 6:

Theorem 2 *(Margin-based generalization bound for ranking) For $\varepsilon > 0$, $\theta > 0$ with probability at least*

$$1 - 2\mathcal{N}\left(\mathcal{F}, \frac{\theta}{4}, \frac{\varepsilon}{8}\right) \exp\left[-\frac{m(\varepsilon E)^2}{8}\right]$$

over the random choice of the training set S, every $f \in \mathcal{F}$ satisfies:

$$\mathbb{P}_{\mathcal{D}}\{\text{misrank}_f\} \leq \mathbb{P}_S\{\text{margin}_f \leq \theta\} + \varepsilon.$$

In other words, the misranking probability is upper bounded by the fraction of instances with margin below θ, plus ε; this statement is true with probability depending on $m, E, \theta, \varepsilon$, and $\mathcal{F}$.

We have chosen to write our bound in terms of E, but we could equally well have used an analogous empirical quantity, namely

$$\mathbb{E}_{\mathbf{x}_i, \mathbf{x}_k \sim S}[\pi(\mathbf{x}_i, \mathbf{x}_k)] = \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{k=1}^{m} \pi(\mathbf{x}_i, \mathbf{x}_k).$$

This is an arbitrary decision; we can in no way influence $\mathbb{E}_{\mathbf{x}_i, \mathbf{x}_k \sim S}[\pi(\mathbf{x}_i, \mathbf{x}_k)]$ in our setting, since we are choosing training instances randomly. E can be viewed as a constant, where recall $0 < E \leq 1/2$. If $E = 0$, it means that there is no information about the relative ranks of examples, and accordingly the bound becomes trivial. Note that in the special bipartite case, E is the proportion of positive examples multiplied by the proportion of negative examples.

In order to see that this bound encourages the margin to be made large, consider the simplified case where the empirical error term is 0, that is, $\mathbb{P}_S\{\text{margin}_f \leq \theta\} = 0$. Now, the only place where

θ appears is in the covering number. In order to make the probability of success larger, the covering number should be made as small as possible, which implies that θ should be made as large as possible.

As a special case of the theorem, we consider the standard setting where f is a (normalized) linear combination of a dictionary of step functions (or "weak rankers"). In this case, we can show the following, proved in Section 6:

Lemma 3 *(Upper bound on covering numbers for convex combinations of binary weak classifiers) For the following hypothesis space:*

$$\mathcal{F} = \left\{ f : f = \sum_j \lambda_j h_j,\ \sum_j \lambda_j = 1,\ \forall j\, \lambda_j \geq 0,\ h_j : \mathcal{X} \to \{0,1\}, h_j \in \mathcal{H} \right\},$$

we have

$$\ln \mathcal{N}(\mathcal{F}, \theta, \varepsilon) \leq \frac{\ln |\mathcal{H}|\ \ln(2/\varepsilon)}{2\theta^2}.$$

Thus, Theorem 2 implies the following corollary.

Corollary 4 *(Margin-based generalization bound for ranking, convex combination of binary weak rankers) For $\varepsilon > 0$, $\theta > 0$ with probability at least*

$$1 - 2\exp\left(\frac{\ln |\mathcal{H}|\ \ln(16/\varepsilon)}{\theta^2/8} - \frac{m(\varepsilon E)^2}{8} \right)$$

over the random choice of the training set S, every $f \in \mathcal{F}$ satisfies:

$$\mathbb{P}_{\mathcal{D}}\{\text{misrank}_f\} \leq \mathbb{P}_S\{\text{margin}_f \leq \theta\} + \varepsilon.$$

In this case, we can lower bound the right hand side by $1 - \delta$ for an appropriate choice of ε. In particular, Corollary 4 implies that

$$\mathbb{P}_{\mathcal{D}}\{\text{misrank}_f\} \leq \mathbb{P}_S\{\text{margin}_f \leq \theta\} + \varepsilon$$

with probability at least $1 - \delta$ if

$$\varepsilon = \sqrt{\frac{4}{mE^2}\left[\frac{8\ln|\mathcal{H}|}{\theta^2} \ln\left(\frac{4mE^2\theta^2}{\ln|\mathcal{H}|} \right) + 2\ln\left(\frac{2}{\delta} \right) \right]}. \tag{2}$$

This bound holds provided that θ is not too small relative to m, specifically, if

$$m\theta^2 \geq \frac{64\ln|\mathcal{H}|}{E^2}.$$

Note that the bound in (2) is only polylogarithmic in $|\mathcal{H}|$.

As we have discussed above, Theorem 2 can be trivially upper bounded using the L_∞ covering number.

Corollary 5 *(Margin-based generalization bound for ranking, L_∞ covering numbers) For $\varepsilon > 0$, $\theta > 0$ with probability at least*

$$1 - 2\mathcal{N}_\infty\left(\mathcal{F}, \frac{\theta}{4}\right)\exp\left[-\frac{m(\varepsilon E)^2}{8}\right]$$

over the random choice of the training set S, every $f \in \mathcal{F}$ satisfies:

$$\mathbb{P}_{\mathcal{D}}\{\text{misrank}_f\} \leq \mathbb{P}_S\{\text{margin}_f \leq \theta\} + \varepsilon.$$

Consider the case of a finite hypothesis space $\mathcal{F}$ where every function is far apart (in L_∞) from every other function. In this case, the covering number is equal to the number of functions. This is the worst possible case, where $\mathcal{N}\left(\mathcal{F}, \frac{\theta}{4}\right) = |\mathcal{F}|$ for any value of θ. In this case, we can solve for ε directly:

$$\delta := 2|\mathcal{F}|\exp\left[-\frac{m(\varepsilon E)^2}{8}\right] \Longrightarrow \varepsilon = \frac{1}{\sqrt{m}}\sqrt{\frac{8}{E^2}(\ln 2|\mathcal{F}| + \ln(1/\delta))}.$$

This indicates that the error may scale as $1/\sqrt{m}$. For the ranking problem, since we are dealing with pairwise relationships, we might expect worse dependence, but this does not appear to be the case. In fact, the dependence on m is quite reasonable in comparison to bounds for the problem of classification, which does not deal with examples pairwise. This is true not only for finite hypothesis spaces (scaling as $1/\sqrt{m}$) but also when the hypotheses are convex combinations of weak rankers (scaling as $\sqrt{\ln(m)/m}$).

4. Coordinate-Based Ranking Algorithms

In the previous section we presented a uniform bound that holds for all $f \in \mathcal{F}$. In this section, we discuss how a learning algorithm might pick one of those functions in order to make $\mathbb{P}_{\mathcal{D}}\{\text{misrank}_f\}$ as small as possible, based on intuition gained from the bound of Theorem 2. Our bound suggests that given a fixed hypothesis space $\mathcal{F}$ and a fixed number of instances m we try to maximize the margin. We will do this using coordinate ascent. Coordinate ascent/descent is similar to gradient ascent/descent except that the optimization moves along single coordinate axes rather than along the gradient. (See Burges et al., 2005, for a gradient-based ranking algorithm based on a probabilistic model.) We first derive the plain coordinate descent version of RankBoost, and show that it is different from RankBoost itself. In Section 4.2 we define the smooth ranking margin $\tilde{G}$. Then we present the "smooth margin ranking" algorithm, and prove that it makes significant progress towards increasing this smooth ranking margin at each iteration, and converges to a maximum margin solution.

4.1 Coordinate Descent and Its Variation on RankBoost's Objective

We take the hypothesis space $\mathcal{F}$ to be the class of convex combinations of weak rankers $\{h_j\}_{j=1,\ldots,n}$, where $h_j : X \to \{0,1\}$. The function f is constructed as a normalized linear combination of the h_j's:

$$f = \frac{\sum_j \lambda_j h_j}{||\boldsymbol{\lambda}||_1},$$

where $||\boldsymbol{\lambda}||_1 = \sum_j \lambda_j, \lambda_j \geq 0$.

We will derive and mention many different algorithms based on different objective functions; here is a summary of them:

$F(\boldsymbol{\lambda})$: For the *classification* problem, AdaBoost minimizes its objective, denoted $F(\boldsymbol{\lambda})$, by coordinate descent.

$G(\boldsymbol{\lambda})$: For *classification limited to the separable case*, the algorithms "coordinate ascent boosting" and "approximate coordinate ascent boosting" are known to maximize the margin (Rudin et al., 2007). These algorithms are based on the smooth classification margin $G(\boldsymbol{\lambda})$.

$\tilde{F}(\boldsymbol{\lambda})$: For *ranking*, "coordinate descent RankBoost" minimizes its objective, denoted $\tilde{F}(\boldsymbol{\lambda})$, by coordinate descent. RankBoost itself minimizes $\tilde{F}(\boldsymbol{\lambda})$ by a variation of coordinate descent that chooses the coordinate with knowledge of the step size.

$\tilde{G}(\boldsymbol{\lambda})$: For *ranking limited to the separable case*, "smooth margin ranking" is an approximate coordinate ascent algorithm that maximizes the ranking margin. It is based on the smooth ranking margin $\tilde{G}(\boldsymbol{\lambda})$.

The objective function for RankBoost is a sum of exponentiated margins:

$$\tilde{F}(\boldsymbol{\lambda}) := \sum_{\{i,k:[\pi(\mathbf{x}_i,\mathbf{x}_k)=1]\}} e^{-\left(\sum_j \lambda_j h_j(\mathbf{x}_i) - \sum_j \lambda_j h_j(\mathbf{x}_k)\right)} = \sum_{ik \in \mathcal{C}_p} e^{-(\mathbf{M}\boldsymbol{\lambda})_{ik}},$$

where we have rewritten in terms of a structure $\mathbf{M}$, which describes how each individual weak ranker j ranks each crucial pair $\mathbf{x}_i, \mathbf{x}_k$; this will make notation significantly easier. Define an index set that enumerates all crucial pairs $\mathcal{C}_p = \{i,k : \pi(\mathbf{x}_i,\mathbf{x}_k) = 1\}$. Formally, the elements of the two-dimensional matrix $\mathbf{M}$ are defined as follows, for index ik corresponding to crucial pair $\mathbf{x}_i, \mathbf{x}_k$:

$$M_{ik,j} := h_j(\mathbf{x}_i) - h_j(\mathbf{x}_k).$$

The first index of $\mathbf{M}$ is ik, which runs over crucial pairs, that is, elements of $\mathcal{C}_p$, and the second index j runs over weak rankers. The size of $\mathbf{M}$ is $|\mathcal{C}_p| \times n$. Since the weak rankers are binary, the entries of $\mathbf{M}$ are within $\{-1, 0, 1\}$. The notation $(\cdot)_j$ means the j^{th} index of the vector, so that the following notation is defined:

$$(\mathbf{M}\boldsymbol{\lambda})_{ik} := \sum_{j=1}^{n} M_{ik,j}\lambda_j = \sum_{j=1}^{n} \lambda_j h_j(\mathbf{x}_i) - \lambda_j h_j(\mathbf{x}_k), \text{ and } (\mathbf{d}^T\mathbf{M})_j := \sum_{ik \in \mathcal{C}_p} d_{ik} M_{ik,j},$$

for $\boldsymbol{\lambda} \in \mathbb{R}^n$ and $\mathbf{d} \in \mathbb{R}^{|\mathcal{C}_p|}$.

4.1.1 COORDINATE DESCENT RANKBOOST

Let us perform standard coordinate descent on this objective function, and we will call the algorithm "coordinate descent RankBoost." We will not get the RankBoost algorithm this way; we will show how to do this in Section 4.1.2. For coordinate descent on $\tilde{F}$, at iteration t, we first choose a direction j_t in which $\tilde{F}$ is decreasing very rapidly. The direction chosen at iteration t (corresponding to the choice of weak ranker j_t) in the "optimal" case (where the best weak ranker is chosen at each iteration) is given as follows. The notation $\mathbf{e}_j$ indicates a vector of zeros with a 1 in the j^{th} entry:

$$\begin{aligned} j_t \in \operatorname*{argmax}_j \left[-\frac{\partial \tilde{F}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_j)}{\partial \alpha}\Big|_{\alpha=0} \right] &= \operatorname*{argmax}_j \sum_{ik \in \mathcal{C}_p} e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}} M_{ik,j} \\ &= \operatorname*{argmax}_j \sum_{ik \in \mathcal{C}_p} d_{t,ik} M_{ik,j} = \operatorname*{argmax}_j (\mathbf{d}_t^T \mathbf{M})_j, \end{aligned} \quad (3)$$

where the "weights" $d_{t,ik}$ are defined by:

$$d_{t,ik} := \frac{e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}}}{\tilde{F}(\boldsymbol{\lambda}_t)} = \frac{e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}}}{\sum_{\tilde{ik}\in C_p} e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{\tilde{ik}}}}.$$

From this calculation, one can see that the chosen weak ranker is a natural choice, namely, j_t is the most accurate weak ranker with respect to the weighted crucial training pairs; maximizing $(\mathbf{d}_t^T\mathbf{M})_j$ encourages the algorithm to choose the most accurate weak ranker with respect to the weights.

The step size our coordinate descent algorithm chooses at iteration t is α_t, where α_t satisfies the following equation for the line search along direction j_t. Define $I_{t+} := \{ik : M_{ik,j_t} = 1\}$, and similarly, $I_{t-} := \{ik : M_{ik,j_t} = -1\}$. Also define $d_{t+} := \sum_{ik\in I_+} d_{t,ik}$ and $d_{t-} := \sum_{ik\in I_-} d_{t,ik}$. The line search is:

$$\begin{aligned}
0 &= -\frac{\partial \tilde{F}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_{j_t})}{\partial \alpha}\Big|_{\alpha=\alpha_t} = \sum_{ik\in C_p} e^{-(\mathbf{M}(\boldsymbol{\lambda}_t+\alpha_t \mathbf{e}_{j_t}))_{ik}} M_{ik,j_t} \\
&= \sum_{ik\in I_{t+}} e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}} e^{-\alpha_t} - \sum_{ik\in I_{t-}} e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}} e^{\alpha_t} \\
0 &= d_{t+} e^{-\alpha_t} - d_{t-} e^{\alpha_t} \\
\alpha_t &= \frac{1}{2}\ln\left(\frac{d_{t+}}{d_{t-}}\right). \qquad (4)
\end{aligned}$$

Thus, we have derived the first algorithm, coordinate descent RankBoost. Pseudocode can be found in Figure 1. In order to make the calculation for $\mathbf{d}_t$ numerically stable, we write $\mathbf{d}_t$ in terms of its update from the previous iteration.

4.1.2 RANKBOOST

Let us contrast coordinate descent RankBoost with RankBoost. They both minimize the same objective $\tilde{F}$, but they differ by the ordering of steps: for coordinate descent RankBoost, j_t is calculated first, then α_t. In contrast, RankBoost uses the formula (4) for α_t in order to calculate j_t. In other words, at each step RankBoost selects the weak ranker that yields the largest decrease in the loss function, whereas coordinate descent RankBoost selects the weak ranker of steepest slope. Let us derive RankBoost. Define the following for iteration t (eliminating the t subscript):

$$\begin{aligned}
I_{+j} &:= \{ik : M_{ik,j} = 1\}, \quad I_{-j} := \{ik : M_{ik,j} = -1\}, \quad I_{0j} := \{ik : M_{ik,j} = 0\}, \\
d_{+j} &:= \sum_{ik\in I_{+j}} d_{t,ik}, \quad d_{-j} := \sum_{ik\in I_{-j}} d_{t,ik}, \quad d_{0j} := \sum_{ik\in I_{0j}} d_{t,ik}.
\end{aligned}$$

For each j, we take a step according to (4) of size $\frac{1}{2}\ln\frac{d_{+j}}{d_{-j}}$, and choose the j_t which makes the objective function $\tilde{F}$ decrease the most. That is:

$$\begin{aligned}
j_t : &= \operatorname*{argmin}_j \tilde{F}\left(\boldsymbol{\lambda}_t + \left(\frac{1}{2}\ln\frac{d_{+j}}{d_{-j}}\right)\mathbf{e}_{j_t}\right) = \operatorname*{argmin}_j \sum_{ik\in C_p} e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}} e^{-M_{ik,j}\frac{1}{2}\ln\frac{d_{+j}}{d_{-j}}} \\
&= \operatorname*{argmin}_j \sum_{ik} d_{t,ik}\left(\frac{d_{+j}}{d_{-j}}\right)^{-\frac{1}{2}M_{ik,j}} \\
&= \operatorname*{argmin}_j \left[2(d_{+j}d_{-j})^{1/2} + d_{0j}\right]. \qquad (5)
\end{aligned}$$

1. **Input:** Matrix $\mathbf{M}$, No. of iterations t_{max}

2. **Initialize:** $\lambda_{1,j} = 0$ for $j = 1,...,n$, $d_{1,ik} = 1/m$ for all ik

3. **Loop for** $t = 1,...,t_{max}$

   (a) $j_t \in \operatorname{argmax}_j (\mathbf{d}_t^T \mathbf{M})_j$ "optimal" case choice of weak classifier

   (b) $d_{t+} = \sum_{\{ik:M_{ik,j_t}=1\}} d_{t,ik}$, $d_{t-} = \sum_{\{ik:M_{ik,j_t}=-1\}} d_{t,ik}$

   (c) $\alpha_t = \frac{1}{2} \ln \left(\frac{d_{t+}}{d_{t-}} \right)$

   (d) $d_{t+1,ik} = d_{t,ik} e^{-M_{ik,j_t} \alpha_t} /$normaliz. for each crucial pair ik in C_p

   (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 coordinate descent RankBoost.

After we make the choice of j_t, then we can plug back into the formula for α_t, yielding $\alpha_t = \frac{1}{2} \ln \frac{d_{+j_t}}{d_{-j_t}}$. We have finished re-deriving RankBoost. As we mentioned before, the plain coordinate descent algorithm has more natural weak learning associated with it, since the weak ranker chosen tries to find the most accurate weak ranker with respect to the weighted crucial pairs; in other words, we argue (3) is a more natural weak learner than (5).

Note that for AdaBoost's objective function, choosing the weak classifier with the steepest slope (plain coordinate descent) yields the same as choosing the weak classifier with the largest decrease in the loss function: both yield AdaBoost.[2]

2. For AdaBoost, entries of the matrix $\mathbf{M}$ are $M_{ij}^{Ada} := y_i h_j(\mathbf{x}_i) \in \{-1,1\}$ since hypotheses are assumed to be $\{-1,1\}$ valued for AdaBoost. Thus $d_{0j} = 0$, and from plain coordinate descent: $j_t = \operatorname{argmax}_j d_{+j} - d_{-j} = \operatorname{argmax}_j 2d_{+j} - 1$, that is, $j_t = \operatorname{argmax}_j d_{+j}$. On the other hand, for the choice of weak classifier with the greatest decreases in the loss (same calculation as above):

$$\begin{aligned} j_t &= \operatorname{argmin}_j 2(d_{+j} d_{-j})^{1/2}, \text{ that is,} \\ j_t &= \operatorname{argmin}_j d_{+j}(1 - d_{+j}) = \operatorname{argmax}_j d_{+j}^2 - d_{+j}, \end{aligned}$$

and since $d_{+j} > 1/2$, the function $d_{+j}^2 - d_{+j}$ is monotonically increasing in d_{+j}, so $j_t = \operatorname{argmax}_j d_{+j}$. Thus, whether or not AdaBoost chooses its weak classifier with knowledge of the step size, it would choose the same weak classifier anyway.

4.2 Smooth Margin Ranking

The value of $\tilde{F}$ does not directly tell us anything about the margin, only whether the margin is positive. In fact, it is possible to minimize $\tilde{F}$ with a positive margin that is arbitrarily small, relative to the optimal.[3] Exactly the same problem occurs for AdaBoost. It has been proven (Rudin et al., 2004) that it is possible for AdaBoost not to converge to a maximum margin solution, nor even to make progress towards increasing the margin at every iteration. Thus, since the calculations are identical for RankBoost, there are certain cases in which we can expect RankBoost not to converge to a maximum margin solution.

Theorem 6 *(RankBoost does not always converge to a maximum margin solution) There exist matrices $\boldsymbol{M}$ for which RankBoost converges to a margin that is strictly less than the maximum margin.*

Proof Since RankBoost and AdaBoost differ only in their definitions of the matrix $\mathbf{M}$, they possess exactly the same convergence properties for the same choice of $\mathbf{M}$. There is an 8×8 matrix $\mathbf{M}$ in Rudin et al. (2004) for which AdaBoost converges to a margin value of $1/3$, when the maximum margin is $3/8$. Thus, the same convergence property applies for RankBoost. It is rare in the separable case to be able to solve for the asymptotic margin that AdaBoost or RankBoost converges to; for this 8×8 example, AdaBoost's weight vectors exhibit cyclic behavior, which allowed convergence of the margin to be completely determined. ■

A more complete characterization of AdaBoost's convergence with respect to the margin (and thus RankBoost's convergence) can be found in Rudin et al. (2007).

In earlier work, we have introduced a smooth margin function, which one can maximize in order to achieve a maximum margin solution for the classification problem (Rudin et al., 2007). A coordinate ascent algorithm on this function makes progress towards increasing the smooth margin at every iteration. Here, we present the analogous smooth ranking function and the smooth margin ranking algorithm. The extension of the convergence proofs for this algorithm is nontrivial; our main contribution in this section is a condition under which the algorithm makes progress.

The smooth ranking function $\tilde{G}$ is defined as follows:

$$\tilde{G}(\boldsymbol{\lambda}) := \frac{-\ln \tilde{F}(\boldsymbol{\lambda})}{||\boldsymbol{\lambda}||_1}.$$

It is not hard to show (see Rudin et al., 2007) that:

$$\tilde{G}(\boldsymbol{\lambda}) < \mu(\boldsymbol{\lambda}) \leq \rho, \tag{6}$$

where the margin can be written in this notation as:

$$\mu(\boldsymbol{\lambda}) = \min_i \frac{(\mathbf{M}\boldsymbol{\lambda})_i}{\|\boldsymbol{\lambda}\|_1}$$

3. One can see this by considering any vector $\boldsymbol{\lambda}$ such that $(\mathbf{M}\boldsymbol{\lambda})_{ik}$ is positive for all crucial pairs ik. That is, we choose any $\boldsymbol{\lambda}$ that yields a positive margin. We can make the value of $\tilde{F}$ arbitrarily small by multiplying $\boldsymbol{\lambda}$ by a large positive constant; this will not affect the value of the margin because the margin is $\min_{ik \in \mathcal{C}_p} (\mathbf{M}\boldsymbol{\lambda})_{ik} / ||\boldsymbol{\lambda}||_1$, and the large constant will cancel. In this way, the objective can be arbitrarily small, while the margin is certainly not maximized. Thus, coordinate descent on $\tilde{F}$ does not necessarily have anything to do with maximizing the margin.

and the best possible margin is:

$$\rho = \min_{\{\mathbf{d}:\sum_{ik} d_{ik}=1, d_{ik}\geq 0\}} \max_j (\mathbf{d}^T \mathbf{M})_j = \max_{\{\bar{\boldsymbol{\lambda}}:\sum_j \bar{\lambda}_j=1, \bar{\lambda}_j\geq 0\}} \min_i (\mathbf{M}\bar{\boldsymbol{\lambda}})_i.$$

In other words, the smooth ranking margin is always less than the true margin, although the two quantities become closer as $||\boldsymbol{\lambda}||_1$ increases. The true margin is no greater than ρ, the min-max value of the game defined by $\mathbf{M}$ (see Freund and Schapire, 1999).

We now define the smooth margin ranking algorithm, which is approximately coordinate ascent on $\tilde{G}$. As usual, the input to the algorithm is matrix $\mathbf{M}$, determined from the training data. Also, we will only define this algorithm when $\tilde{G}(\boldsymbol{\lambda})$ is positive, so that we only use it once the data has become separable; we can use RankBoost or coordinate descent RankBoost to get us to this point.

We will define iteration $t+1$ in terms of the quantities known at iteration t. At iteration t, we have calculated $\boldsymbol{\lambda}_t$, at which point the following quantities can be calculated:

$$\begin{aligned} g_t &:= \tilde{G}(\boldsymbol{\lambda}_t) \\ \text{weights on crucial pairs } d_{t,ik} &:= e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}}/\tilde{F}(\boldsymbol{\lambda}_t) \\ \text{direction } j_t &= \operatorname*{argmax}_j (\mathbf{d}_t^T \mathbf{M})_j \\ \text{edge } r_t &:= (\mathbf{d}_t^T \mathbf{M})_{j_t}. \end{aligned}$$

The choice of j_t is the same as for coordinate descent RankBoost (also see Rudin et al., 2007). The step size α_t is chosen to obey Equation (12) below, but we need a few more definitions before we state its value, so we do not define it yet; we will first define recursive equations for $\tilde{F}$ and $\tilde{G}$. We also have $s_t = ||\boldsymbol{\lambda}_t||_1$ and $s_{t+1} = s_t + \alpha_t$, and $g_{t+1} = \tilde{G}(\boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t})$, where α_t has not yet been defined.

As before, $I_{t+} := \{i,k | M_{ikj_t} = 1, \pi(\mathbf{x}_i, \mathbf{x}_k) = 1\}$, $I_{t-} := \{i,k | M_{ikj_t} = -1, \pi(\mathbf{x}_i, \mathbf{x}_k) = 1\}$, and now, $I_{t0} := \{i,k | M_{ikj_t} = 0, \pi(\mathbf{x}_i, \mathbf{x}_k) = 1\}$. Also $d_{t+} := \sum_{I_{t+}} d_{t,ik}$, $d_{t-} := \sum_{I_{t-}} d_{t,ik}$, and $d_{t0} := \sum_{I_{t0}} d_{t,ik}$. Thus, by definition, we have $d_{t+} + d_{t-} + d_{t0} = 1$. Now, r_t can be written $r_t = d_{t+} - d_{t-}$. Define the factor

$$\tau_t := d_{t+} e^{-\alpha_t} + d_{t-} e^{\alpha_t} + d_{t0}, \tag{7}$$

and define its "derivative":

$$\tau_t' := \frac{\partial \tau_t (d_{t+} e^{-\alpha} + d_{t-} e^{\alpha} + d_{t0})}{\partial \alpha}\Big|_{\alpha=\alpha_t} = -d_{t+} e^{-\alpha_t} + d_{t-} e^{\alpha_t}. \tag{8}$$

We now derive a recursive equation for $\tilde{F}$, true for any α.

$$\begin{aligned} \tilde{F}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_{j_t}) &= \sum_{\{i,k|\pi(\mathbf{x}_i,\mathbf{x}_k)=1\}} e^{(-\mathbf{M}\boldsymbol{\lambda}_t)_{ik}} e^{-M_{ikj_t}\alpha} \\ &= \tilde{F}(\boldsymbol{\lambda}_t)(d_{t+} e^{-\alpha} + d_{t-} e^{\alpha} + d_{t0}). \end{aligned}$$

Thus, we have defined τ_t so that

$$\tilde{F}(\boldsymbol{\lambda}_{t+1}) = \tilde{F}(\boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t}) = \tilde{F}(\boldsymbol{\lambda}_t)\tau_t.$$

We use this to write a recursive equation for $\tilde{G}$.

$$\begin{aligned}\tilde{G}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_{j_t}) &= \frac{-\ln(\tilde{F}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_{j_t}))}{s_t + \alpha} = \frac{-\ln(\tilde{F}(\boldsymbol{\lambda}_t)) - \ln\left(d_{t+}e^{-\alpha} + d_{t-}e^{\alpha} + d_{t0}\right)}{s_t + \alpha} \\ &= g_t \frac{s_t}{s_t + \alpha} - \frac{\ln\left(d_{t+}e^{-\alpha} + d_{t-}e^{\alpha} + d_{t0}\right)}{s_t + \alpha}.\end{aligned}$$

For our algorithm, we set $\alpha = \alpha_t$ in the above expression and use the notation defined earlier:

$$\begin{aligned}g_{t+1} &= g_t \frac{s_t}{s_t + \alpha_t} - \frac{\ln \tau_t}{s_t + \alpha_t} \\ g_{t+1} - g_t &= \frac{g_t s_t - g_t s_t - g_t \alpha_t}{s_t + \alpha_t} - \frac{\ln \tau_t}{s_t + \alpha_t} = -\frac{1}{s_{t+1}} \left[g_t \alpha_t + \ln \tau_t\right]. \end{aligned} \tag{9}$$

Now we have gathered enough notation to write the equation for α_t for smooth margin ranking. For plain coordinate ascent, the update α^* solves:

$$\begin{aligned}0 &= \frac{\partial \tilde{G}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_{j_t})}{\partial \alpha}\Big|_{\alpha=\alpha^*} = \frac{\partial}{\partial \alpha}\left[\frac{-\ln \tilde{F}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_{j_t})}{s_t + \alpha}\right]\Big|_{\alpha=\alpha^*} \\ &= \frac{1}{s_t + \alpha^*}\left[-\left[\frac{-\ln \tilde{F}(\boldsymbol{\lambda}_t + \alpha^* \mathbf{e}_{j_t})}{s_t + \alpha^*}\right] + \left[\frac{-\partial \tilde{F}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_{j_t})/\partial \alpha\Big|_{\alpha=\alpha^*}}{\tilde{F}(\boldsymbol{\lambda}_t + \alpha^* \mathbf{e}_{j_t})}\right]\right] \\ &= \frac{1}{s_t + \alpha^*}\left[-\tilde{G}(\boldsymbol{\lambda}_t + \alpha^* \mathbf{e}_{j_t}) + \left[\frac{-\partial \tilde{F}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_{j_t})/\partial \alpha\Big|_{\alpha=\alpha^*}}{\tilde{F}(\boldsymbol{\lambda}_t + \alpha^* \mathbf{e}_{j_t})}\right]\right].\end{aligned} \tag{10}$$

We could solve this equation numerically for α^* to get a smooth margin coordinate ascent algorithm; however, we avoid this line search for α^* in smooth margin ranking. We will do an approximation that allows us to solve for α^* directly so that the algorithm is just as easy to implement as RankBoost. To get the update rule for smooth margin ranking, we set α_t to solve:

$$\begin{aligned}0 &= \frac{1}{s_t + \alpha_t}\left[-\tilde{G}(\boldsymbol{\lambda}_t) + \left[\frac{-\partial \tilde{F}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_{j_t})/\partial \alpha\Big|_{\alpha=\alpha_t}}{\tilde{F}(\boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t})}\right]\right] \\ &= \frac{1}{s_t + \alpha_t}\left(-g_t + \frac{-\tau_t' \tilde{F}(\boldsymbol{\lambda}_t)}{\tau_t \tilde{F}(\boldsymbol{\lambda}_t)}\right) \\ g_t \tau_t &= -\tau_t'.\end{aligned} \tag{11}$$

This expression can be solved analytically for α_t, but we avoid using the exact expression in our calculations whenever possible, since the solution is not that easy to work with in our analysis:

$$\alpha_t = \ln\left[\frac{-g_t d_{t0} + \sqrt{g_t^2 d_{t0}^2 + (1+g_t)(1-g_t)4 d_{t+} d_{t-}}}{(1+g_t)2 d_{t-}}\right]. \tag{12}$$

We are done defining the algorithm and in the process we have derived some useful recursive relationships. In summary:

Smooth margin ranking is the same as described in Figure 1, except that (3c) is replaced by (12), where $d_{t0} = 1 - d_{t+} - d_{t-}$ *and* $g_t = G(\boldsymbol{\lambda}_t)$.

Binary weak rankers were required to obtain an analytical solution for α_t, but if one is willing to perform a 1-dimensional linesearch (10) at each iteration, real-valued features can just as easily be used.

Now we move onto the convergence proofs, which were loosely inspired by the analysis of Zhang and Yu (2005). The following theorem gives conditions when the algorithm makes significant progress towards increasing the value of $\tilde{G}$ at iteration t. An analogous statement was an essential tool for proving convergence properties of approximate coordinate ascent boosting (Rudin et al., 2007), although the proof of the following theorem is significantly more difficult since we could not use the hyperbolic trigonometric tricks from prior work. As usual, the weak learning algorithm must always achieve an edge r_t of at least ρ for the calculation to hold, where recall $r_t = (\mathbf{d}_t^T \mathbf{M})_{j_t} = d_{t+} - d_{t-}$. At every iteration, there is always a weak ranker which achieves edge at least ρ, so this requirement is always met in the "optimal case," where we choose the best possible weak ranker at every iteration (i.e., the argmax over j). There is one more condition in order for the algorithm to make progress, namely that most of the weight should indicate the strength of the weak ranker, which implies that d_{t0} cannot take too much of the weight. Specifically, $d_{t0} < \frac{2}{3}(1-r_t)(1-r_t^2)$, which is derived from a bound on the second derivative of the step size.

Theorem 7 *(Progress according to the smooth margin) For* $0 \leq g_t < r_t < 1$ *and* $0 \leq d_{t0} < \frac{2}{3}(1 - r_t)(1-r_t^2)$ *the algorithm makes progress at iteration t:*

$$g_{t+1} - g_t \geq \frac{1}{2} \frac{\alpha_t (r_t - g_t)}{s_{t+1}}.$$

The proof of this theorem is in Section 7. This theorem tells us that the value of the smooth ranking margin increases significantly when the condition on d_0 holds. This theorem is the main step in proving convergence theorems, for example:

Theorem 8 *(Convergence for smooth margin ranking) If* $d_{t0} < \frac{2}{3}(1-r_t)(1-r_t^2)$ *for all t, the smooth margin ranking algorithm converges to a maximum margin solution, that is,* $\lim_{t\to\infty} g_t = \rho$. *Thus the limiting margin is* ρ, *that is,* $\lim_{t\to\infty} \mu(\boldsymbol{\lambda}_t) = \rho$.

Besides Theorem 7, the only other key step in the proof of Theorem 8 is the following lemma, proved in Section 7:

Lemma 9 *(Step-size does not increase too quickly for smooth margin ranking)*

$$\lim_{t\to\infty} \frac{\alpha_t}{s_{t+1}} = 0.$$

From here, the proof of the convergence theorem is not difficult. The two conditions found in Theorem 7 and Lemma 9 are identical to those of Lemma 5.1 and Lemma 5.2 of Rudin et al. (2007). These are the only two ingredients necessary to prove asymptotic convergence using the proof outline of Theorem 5.1 of Rudin et al. (2007); an adaptation of this proof suffices to show Theorem 8, which we now outline.

Proof *(of Theorem 8)* The values of g_t constitute a nondecreasing sequence which is uniformly bounded by 1. Thus, a limit g_∞ must exist, $g_\infty := \lim_{t\to\infty} g_t$. By (6), we know that $g_t \leq \rho$ for all

t. Thus, $g_\infty \leq \rho$. Let us suppose that $g_\infty < \rho$, so that $\rho - g_\infty \neq 0$. This assumption, together with Theorem 7 and Lemma 9 can be used in the same way as in Rudin et al. (2007) to show that $\sum_t \alpha_t$ is finite, implying that:

$$\lim_{t\to\infty} \alpha_t = 0.$$

Using this fact along with (11), we find:

$$\begin{aligned} g_\infty &= \lim_{t\to\infty} g_t = \liminf_{t\to\infty} g_t = \liminf_{t\to\infty} \frac{-\tau_t'}{\tau_t} = \liminf_{t\to\infty} \frac{-(-d_{t+}e^{-\alpha_t} + d_{t-}e^{\alpha_t})}{d_{t+}e^{-\alpha_t} + d_{t-}e^{\alpha_t} + d_{t0}} \\ &= \liminf_{t\to\infty} r_t \geq \rho. \end{aligned}$$

This is a contradiction with the original assumption that $g_\infty < \rho$. It follows that $g_\infty = \rho$, or $\lim_{t\to\infty}(\rho - g_t) = 0$. Thus, the smooth ranking algorithm converges to a maximum margin solution. ■

5. AdaBoost and RankBoost in the Bipartite Ranking Problem

In this section, we present an equivalence between AdaBoost and RankBoost in terms of their behavior on the training set. Namely, we show that under very natural conditions, AdaBoost asymptotically produces an area under the ROC curve value that is equally as good as RankBoost's. Conversely, RankBoost (but with a change in the intercept), produces a classification that is equally as good as AdaBoost's. Note that this result is designed for the non-separable case; it holds in the separable case, but the result is trivial since the area under the curve is exactly one. Also, let us be clear that the result is a theoretical proof based on the optimization of the training set only. It is not an experimental result, nor is it a probabilistic guarantee about performance on a test set (such as Theorem 2).

In the bipartite ranking problem, the focus of this section, recall that every training instance falls into one of two categories, the positive class Y_+ and the negative class Y_-. We will take $\pi(\mathbf{x}_i, \mathbf{x}_k) = 1$ for each pair $i \in Y_+$ and $k \in Y_-$ so that crucial pairs exist between elements of the positive class and elements of the negative class. Define $y_i = +1$ when $i \in Y_+$, and $y_i = -1$ otherwise. The AUC (area under the Receiver Operator Characteristic curve) is equivalent to the Mann-Whitney U statistic, and it is closely related to the fraction of misranks. Specifically,

$$1 - \text{AUC}(\boldsymbol{\lambda}) = \frac{\sum_{i\in Y_+}\sum_{k\in Y_-} \mathbf{1}_{[(\mathbf{M}\boldsymbol{\lambda})_{ik}\leq 0]}}{|Y_+||Y_-|} = \text{fraction of misranks.}$$

In the bipartite ranking problem, the function $\tilde{F}$ becomes an exponentiated version of the AUC, that is, since $\mathbf{1}_{[x\leq 0]} \leq e^{-x}$, we have:

$$|Y_+||Y_-|(1 - \text{AUC}(\boldsymbol{\lambda})) = \sum_{i\in Y_+}\sum_{k\in Y_-} \mathbf{1}_{[(\mathbf{M}\boldsymbol{\lambda})_{ik}\leq 0]} \leq \sum_{i\in Y_+}\sum_{k\in Y_-} e^{-(\mathbf{M}\boldsymbol{\lambda})_{ik}} = \tilde{F}(\boldsymbol{\lambda}). \tag{13}$$

We define the matrix $\mathbf{M}^{Ada}$, which is helpful for describing AdaBoost. $\mathbf{M}^{Ada}$ is defined elementwise by $M^{Ada}_{ij} = y_i h_j(\mathbf{x}_i)$ for $i = 1,...,m$ and $j = 1,...,n$. Thus, $M_{ikj} = h_j(\mathbf{x}_i) - h_j(\mathbf{x}_k) = y_i h_j(\mathbf{x}_i) + y_k h_j(\mathbf{x}_k) = M^{Ada}_{ij} + M^{Ada}_{kj}$. (To change from AdaBoost's usual $\{-1,1\}$ hypotheses to RankBoost's usual $\{0,1\}$ hypotheses, divide entries of $\mathbf{M}$ by 2.) Define the following functions:

$$F_+(\boldsymbol{\lambda}) := \sum_{i\in Y_+} e^{-(\mathbf{M}^{Ada}\boldsymbol{\lambda})_i} \text{ and } F_-(\boldsymbol{\lambda}) := \sum_{k\in Y_-} e^{-(\mathbf{M}^{Ada}\boldsymbol{\lambda})_k}.$$

The objective function for AdaBoost is $F(\boldsymbol{\lambda}) := F_+(\boldsymbol{\lambda}) + F_-(\boldsymbol{\lambda})$. The objective function for RankBoost is:

$$\begin{aligned}\tilde{F}(\boldsymbol{\lambda}) &= \sum_{i\in Y_+}\sum_{k\in Y_-}\exp\left[-\sum_j \lambda_j h_j(\mathbf{x}_i)\right]\exp\left[+\sum_j \lambda_j h_j(\mathbf{x}_k)\right] \\ &= \sum_{i\in Y_+}\sum_{k\in Y_-}\exp\left[-\sum_j \lambda_j y_i h_j(\mathbf{x}_i)\right]\exp\left[-\sum_j \lambda_j y_k h_j(\mathbf{x}_k)\right] = F_+(\boldsymbol{\lambda})F_-(\boldsymbol{\lambda}). \qquad (14)\end{aligned}$$

Thus, both objective functions involve exponents of the margins of the training instances, but with a different balance between the positive and negative instances. In both cases, the objective function favors instances to be farther away from the decision boundary—even when the instances are correctly classified and not close to the decision boundary. (This is in contrast to support vector machines which do not suffer any loss for non-support vectors. This is the main reason why an analogous result does not hold for SVMs.)

We now define a quantity called *F-skew*:

$$\text{F-skew}(\boldsymbol{\lambda}) := F_+(\boldsymbol{\lambda}) - F_-(\boldsymbol{\lambda}). \qquad (15)$$

F-skew is the exponentiated version of the "skew," which measures the imbalance between positive and negative instances. The "skew" plays an important role in the expressions of Cortes and Mohri (2004, 2005) and Agarwal et al. (2005). The F-skew measures how much greater the positive instances contribute to AdaBoost's objective than the negative instances. If the F-skew is 0, it means that the positive and negative classes are contributing equally.

The following theorem shows that whenever the F-skew vanishes, any sequence $\boldsymbol{\lambda}_t$ that optimizes AdaBoost's objective F also optimizes RankBoost's objective $\tilde{F}$, and vice versa.

Theorem 10 *(Equivalence between AdaBoost and RankBoost's objectives) Let $\{\boldsymbol{\lambda}_t\}_{t=1}^{\infty}$ be any sequence for which AdaBoost's objective is minimized,*

$$\lim_{t\to\infty} F(\boldsymbol{\lambda}_t) = \inf_{\boldsymbol{\lambda}} F(\boldsymbol{\lambda}), \qquad (16)$$

and $\lim_{t\to\infty} \text{F-skew}(\boldsymbol{\lambda}_t) = 0$. *Then RankBoost's objective is minimized,*

$$\lim_{t\to\infty} \tilde{F}(\boldsymbol{\lambda}_t) = \inf_{\boldsymbol{\lambda}} \tilde{F}(\boldsymbol{\lambda}). \qquad (17)$$

Conversely, for any sequence for which RankBoost's objective is minimized, and for which the F-skew vanishes, AdaBoost's objective is minimized as well.

The proof of the converse follows directly from

$$(F_+(\boldsymbol{\lambda}) + F_-(\boldsymbol{\lambda}))^2 - (F_+(\boldsymbol{\lambda}) - F_-(\boldsymbol{\lambda}))^2 = 4F_+(\boldsymbol{\lambda})F_-(\boldsymbol{\lambda}),$$

Equations (14) and (15), and continuity of the functions involved. The proof of the forward direction in Section 8 uses a theory of convex duality for Bregman divergences developed by Della Pietra et al. (2002) and used by Collins et al. (2002). This theory allows characterization for functions that may have minima at infinity like F and $\tilde{F}$.

Theorem 10 has very practical implications due to the following, proved in Section 8:

Corollary 11 *(AdaBoost minimizes RankBoost's objective) If the constant weak hypothesis $h_0(\mathbf{x}) = 1$ is included in the set of AdaBoost's weak classifiers, or equivalently, if $\mathbf{M}^{Ada}$ has a column j_0 such that $M^{Ada}_{i,j_0} = y_i$ for all i, and if the $\{\boldsymbol{\lambda}_t\}_{t=1}^{\infty}$ sequence obeys (16), then* $\lim_{t\to\infty} \text{F-skew}(\boldsymbol{\lambda}_t) = 0$.

This result and the previous together imply that if the constant weak hypothesis is included in the set of AdaBoost's weak classifiers, then the F-skew vanishes, and RankBoost's objective $\tilde{F}$ is minimized.

Not only does AdaBoost minimize RankBoost's exponential objective function in this case, it also achieves an equally good misranking loss. Before we state this formally as a theorem, we need to avoid a very particular nonuniqueness problem. Namely, there is some ambiguity in the definition of the ranking loss for RankBoost and AdaBoost due to the arbitrariness in the algorithms, and the discontinuity of the function $\mathbf{1}_{[z\leq 0]}$, which is used for the misranking loss $\sum_i \mathbf{1}_{[(\mathbf{M}\boldsymbol{\lambda})_i \leq 0]}$. The arbitrariness in the algorithms arises from the argmax step; since argmax is a set that may contain more than one element, and since the algorithm does not specify which element in that set to choose, solutions might be different for different implementations. There are many examples where the argmax set does contain more than one element (for instance, the examples in Rudin et al., 2004). The vector $\lim_{t\to\infty} \mathbf{1}_{[\mathbf{M}\boldsymbol{\lambda}_t \leq 0]}$ may not be uniquely defined; for some i,k pair we may have $\lim_{t\to\infty} (\mathbf{M}\boldsymbol{\lambda}_t)_{ik} = 0$, and in that case, values of $\lim_{t\to\infty} \mathbf{1}_{[(\mathbf{M}\boldsymbol{\lambda}_t)_{ik} \leq 0]}$ may take on the values 0, 1, or the limit may not exist, depending on the algorithm. Thus, in order to write a sensible theorem, we must eliminate this pathological case. No matter which implementation we choose, this only becomes a problem if $\lim_{t\to\infty} (\mathbf{M}\boldsymbol{\lambda}_t)_{ik} = 0$, that is, there is a tie in the rankings. If there is no tie, the result is deterministic. In other words, when the pathological case is eliminated, the limiting AUC can be defined and AdaBoost asymptotically achieves the same AUC as RankBoost:

Theorem 12 *(AdaBoost and RankBoost achieve the same area under the ROC curve) Consider any two sequences $\{\boldsymbol{\lambda}_t\}_t$ and $\{\boldsymbol{\lambda}'_t\}_t$ that minimize RankBoost's objective $\tilde{F}$, that is,*

$$\lim_{t\to\infty} \tilde{F}(\boldsymbol{\lambda}_t) = \lim_{t\to\infty} \tilde{F}(\boldsymbol{\lambda}'_t) = \inf_{\boldsymbol{\lambda}} \tilde{F}(\boldsymbol{\lambda}).$$

Then, if each positive example has a final score distinct from each negative example, that is, $\forall ik$, $\lim_{t\to\infty} (\boldsymbol{M}\boldsymbol{\lambda}_t)_{ik} \neq 0$, $\lim_{t\to\infty} (\boldsymbol{M}\boldsymbol{\lambda}'_t)_{ik} \neq 0$, then both sequences will asymptotically achieve the same AUC value. That is:

$$\lim_{t\to\infty} \left[\sum_{i\in Y_+} \sum_{k\in Y_-} \mathbf{1}_{[(\mathbf{M}\boldsymbol{\lambda}_t)_{ik} \leq 0]} \right] = \lim_{t\to\infty} \left[\sum_{i\in Y_+} \sum_{k\in Y_-} \mathbf{1}_{[(\mathbf{M}\boldsymbol{\lambda}'_t)_{ik} \leq 0]} \right].$$

The proof is in Section 8. This theorem shows that, in the case where the F-skew vanishes and there are no ties, AdaBoost will generate the same area under the curve value that RankBoost does. That is, a sequence of $\boldsymbol{\lambda}'_t$'s generated by AdaBoost and a sequence of $\boldsymbol{\lambda}_t$'s generated by RankBoost will asymptotically produce the same value of the AUC.

Combining Theorem 10, Corollary 11 and Theorem 12, we can conclude the following, assuming distinct final scores: *if the constant hypothesis is included in the set of AdaBoost's weak classifiers, then AdaBoost will converge to exactly the same area under the ROC curve value as RankBoost*. Given these results, it is now understandable (but perhaps still surprising) that AdaBoost performs so well as a ranking algorithm.

This logic can be made to work in reverse, so that adding a constant hypothesis to RankBoost's output will also produce a minimizer of AdaBoost's objective. In order for this to work, we need to assign the coefficient for the constant classifier (the intercept) to force the F-skew to vanish. Changing the coefficient of the constant hypothesis does not affect RankBoost's objective, but it does affect AdaBoost's. We choose the coefficient to obey the following:

Corollary 13 *(RankBoost minimizes AdaBoost's objective) Define j_0 as the entry corresponding to the constant weak classifier. Take $\boldsymbol{\lambda}_t$ to be a minimizing sequence for RankBoost's objective, that is, $\boldsymbol{\lambda}_t$ obeys (17). Consider $\boldsymbol{\lambda}_t^{corrected}$ where:*

$$\boldsymbol{\lambda}_t^{corrected} := \boldsymbol{\lambda}_t + b_t \mathbf{e}_{j_0},$$

where $\mathbf{e}_{j_0}$ is 1 in the j_0^{th} entry corresponding to the constant weak classifier, and 0 otherwise, and where:

$$b_t = \frac{1}{2} \ln \frac{F_+(\boldsymbol{\lambda}_t)}{F_-(\boldsymbol{\lambda}_t)}.$$

Then, $\boldsymbol{\lambda}_t^{corrected}$ converges to a minimum of AdaBoost's objective, that is, $\boldsymbol{\lambda}_t^{corrected}$ obeys (16).

The proof is in Section 8. Now, we can extend to the misclassification error. The proof of the following is also in Section 8:

Theorem 14 *(AdaBoost and RankBoost achieve the same misclassification error) Consider any two sequences $\{\boldsymbol{\lambda}_t^{corrected}\}_t$ and $\{\boldsymbol{\lambda}_t^{\prime corrected}\}_t$, corrected as in Corollary 13, that minimize RankBoost's objective $\tilde{F}$, that is,*

$$\lim_{t\to\infty} \tilde{F}(\boldsymbol{\lambda}_t^{corrected}) = \lim_{t\to\infty} \tilde{F}(\boldsymbol{\lambda}_t^{\prime corrected}) = \inf_{\boldsymbol{\lambda}} \tilde{F}(\boldsymbol{\lambda}).$$

Then, if no example is on the decision boundary, that is, $\forall i, \lim_{t\to\infty} (\boldsymbol{M}^{Ada}\boldsymbol{\lambda}_t^{corrected})_i \neq 0$, $\forall k \lim_{t\to\infty} (\boldsymbol{M}^{Ada}\boldsymbol{\lambda}_t^{corrected})_k \neq 0$, and $\forall i, \lim_{t\to\infty} (\boldsymbol{M}^{Ada}\boldsymbol{\lambda}_t^{\prime corrected})_i \neq 0, \forall k \lim_{t\to\infty} (\boldsymbol{M}^{Ada}\boldsymbol{\lambda}_t^{\prime corrected})_k \neq 0$, then both sequences will asymptotically achieve the same misclassification loss. That is:

$$\lim_{t\to\infty} \left[\sum_{i\in Y_+} \mathbf{1}_{[(\mathbf{M}^{\mathrm{Ada}}\boldsymbol{\lambda}_t^{corrected})_i \leq 0]} + \sum_{k\in Y_-} \mathbf{1}_{[(\mathbf{M}^{\mathrm{Ada}}\boldsymbol{\lambda}_t^{corrected})_k \leq 0]} \right]$$
$$= \lim_{t\to\infty} \left[\sum_{i\in Y_+} \mathbf{1}_{[(\mathbf{M}^{\mathrm{Ada}}\boldsymbol{\lambda}_t^{\prime corrected})_i \leq 0]} + \sum_{k\in Y_-} \mathbf{1}_{[(\mathbf{M}^{\mathrm{Ada}}\boldsymbol{\lambda}_t^{\prime corrected})_k \leq 0]} \right].$$

Thus, we have shown quite a strong equivalence relationship between RankBoost and AdaBoost. Under natural conditions, AdaBoost achieves the same area under the ROC curve as RankBoost, and RankBoost can be easily made to achieve the same misclassification error as AdaBoost on the training set.

The success of an algorithm is often judged using both misclassification error and the area under the ROC curve. A practical implication of this result is that AdaBoost and RankBoost both solve the classification and ranking problems at the same time. This is true under the conditions specified, namely using a set of binary weak classifiers that includes the constant classifier, and using the

correction for RankBoost's intercept. In terms of which should be used, we have found that AdaBoost tends to converge faster for classification (and uses less memory), whereas RankBoost tends to converge faster for ranking. If the algorithm is stopped early, we suggest that if misclassification error is more important, to choose AdaBoost, and conversely, if area under the ROC curve is more important, to choose RankBoost. Asymptotically, as we have shown, they produce equally good solutions for both classification and ranking on the training set.

5.1 Connection to Multiclass/Multilabel Algorithms

The results above imply convergence properties of two algorithms for solving multiclass/ multilabel problems. Specifically, the algorithms AdaBoost.MH and AdaBoost.MR of Schapire and Singer (1999) have the same relationship to each other as AdaBoost and RankBoost.

In the multilabel setting, each training instance $\mathbf{x} \in \mathcal{X}$ may belong to multiple labels in $\mathcal{Y}$, where $\mathcal{Y}$ is a finite set of labels or classes. The total number of classes is denoted by c. Examples are ordered pairs $(\mathbf{x}, Y), Y \subset \mathcal{Y}$. We use the reduction of Schapire and Singer (1999) where training example i is replaced by a set of single-labeled training examples $\{(\mathbf{x}_i, y_{i\ell})\}_{\ell=1,\ldots,c}$, where $y_{i\ell} = 1$ if $y_{i\ell} \in Y_i$ and -1 otherwise. Thus, the set of training examples are indexed by pairs i, ℓ. Within this reduction, the weak classifiers become $h_j : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.

Let us now re-index the training pairs. The training pairs i, ℓ will now be assigned a single index. Define the entries of matrix $\breve{\mathbf{M}}$ by $\breve{M}_{\breve{\imath}j} = y_{\breve{\imath}} h_j(\mathbf{x}_{\breve{\imath}}, y_{\breve{\imath}})$ for all pairs i, ℓ indexed by $\breve{\imath}$. With this notation, the objective function of AdaBoost.MH becomes:

$$F_{MH}(\boldsymbol{\lambda}) := \sum_{\breve{\imath}} \exp(-\breve{\mathbf{M}}\boldsymbol{\lambda})_{\breve{\imath}}.$$

Using similar notation, the objective function of AdaBoost.MR becomes:

$$F_{MR}(\boldsymbol{\lambda}) := \sum_{\breve{\imath} \in \{\{i,\ell\}: y_{i\ell}=1\}} \exp(-\breve{\mathbf{M}}\boldsymbol{\lambda})_{\breve{\imath}} \sum_{\breve{k} \in \{\{i,\ell\}: y_{i\ell}=-1\}} \exp(-\breve{\mathbf{M}}\boldsymbol{\lambda})_{\breve{k}}.$$

The forms of functions F_{MH} and F_{MR} are the same as those of AdaBoost and RankBoost, respectively, allowing us to directly apply all of the above results. In other words, the same equivalence relationship that we have shown for AdaBoost and RankBoost applies to AdaBoost.MH and AdaBoost.MR.

Now, we move onto the proofs.

6. Proofs from Section 3

This proof in large part follows the approach of Bartlett (1998) and Schapire et al. (1998).

For $f \in \mathcal{F}$, we will be interested in the expectation

$$\mathrm{P}_{\theta,f} := \mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}} \sim \mathcal{D}}\left[f(\bar{\mathbf{x}}) - f(\tilde{\mathbf{x}}) \leq \theta \mid \pi(\bar{\mathbf{x}},\tilde{\mathbf{x}}) = 1\right] = \mathbb{E}_{\bar{\mathbf{x}},\tilde{\mathbf{x}} \sim \mathcal{D}}\left[\mathbf{1}_{[f(\bar{\mathbf{x}}) - f(\tilde{\mathbf{x}}) \leq \theta]} \mid \pi(\bar{\mathbf{x}},\tilde{\mathbf{x}}) = 1\right]$$

as well as its empirical analog

$$\begin{aligned} \hat{\mathrm{P}}_{\theta,f} := \mathbb{P}_S\{\mathrm{margin}_f \leq \theta\} &= \mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}} \sim S}\left[f(\bar{\mathbf{x}}) - f(\tilde{\mathbf{x}}) \leq \theta \mid \pi(\bar{\mathbf{x}},\tilde{\mathbf{x}}) = 1\right] \\ &= \mathbb{E}_{\bar{\mathbf{x}},\tilde{\mathbf{x}} \sim S}\left[\mathbf{1}_{[f(\bar{\mathbf{x}}) - f(\tilde{\mathbf{x}}) \leq \theta]} \mid \pi(\bar{\mathbf{x}},\tilde{\mathbf{x}}) = 1\right]. \end{aligned}$$

Note that in this notation,

$$\mathbb{P}_{\mathcal{D}}\{\text{misrank}_f\} = \mathrm{P}_{0,f}.$$

Our goal is to show that $\mathrm{P}_{0,f} \leq \hat{\mathrm{P}}_{\theta,f} + \varepsilon$ for all $f \in \mathcal{F}$ with high probability. To do so, we will first show that for every $f \in \mathcal{F}$,

$$\mathrm{P}_{0,f} - \hat{\mathrm{P}}_{\theta,f} \leq \mathrm{P}_{\theta/2,g} - \hat{\mathrm{P}}_{\theta/2,g} + \frac{\varepsilon}{2}$$

for some g in the cover $\mathcal{G}$, and then show that the difference $\mathrm{P}_{\theta/2,g} - \hat{\mathrm{P}}_{\theta/2,g}$ on the right must be small for all $g \in \mathcal{G}$, with high probability.

Lemma 15 *Let f and g be any functions in $\mathcal{F}$, and let D be any joint distribution on pairs $\bar{\mathbf{x}}, \tilde{\mathbf{x}}$. Let $0 \leq \theta_1 < \theta_2$. Then*

$$\begin{aligned}
&\mathbb{E}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim D}\left[\mathbf{1}_{[f(\bar{\mathbf{x}})-f(\tilde{\mathbf{x}})\leq\theta_1]} - \mathbf{1}_{[g(\bar{\mathbf{x}})-g(\tilde{\mathbf{x}})\leq\theta_2]}\right] \\
&\quad\leq\quad \mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim D}\left\{|f(\bar{\mathbf{x}}) - g(\bar{\mathbf{x}})| \geq \frac{\theta_2-\theta_1}{2}\right\} + \mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim D}\left\{|f(\tilde{\mathbf{x}}) - g(\tilde{\mathbf{x}})| \geq \frac{\theta_2-\theta_1}{2}\right\}.
\end{aligned}$$

Proof First, note that

$$\mathbf{1}_{[y\leq\theta_1]} - \mathbf{1}_{[z\leq\theta_2]} = \begin{cases} 1 & \text{if } y \leq \theta_1 < \theta_2 < z \\ 0 & \text{otherwise} \end{cases}$$

which means that this difference can be equal to 1 only if $z - y$ is at least $\theta_2 - \theta_1$. Thus,

$$\begin{aligned}
&\mathbb{E}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim D}\left[\mathbf{1}_{[f(\bar{\mathbf{x}})-f(\tilde{\mathbf{x}})\leq\theta_1]} - \mathbf{1}_{[g(\bar{\mathbf{x}})-g(\tilde{\mathbf{x}})\leq\theta_2]}\right] \\
&\quad=\quad \mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim D}\{f(\bar{\mathbf{x}}) - f(\tilde{\mathbf{x}}) \leq \theta_1 < \theta_2 < g(\bar{\mathbf{x}}) - g(\tilde{\mathbf{x}})\} \\
&\quad\leq\quad \mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim D}\{|(f(\bar{\mathbf{x}}) - f(\tilde{\mathbf{x}})) - (g(\bar{\mathbf{x}}) - g(\tilde{\mathbf{x}}))| \geq \theta_2 - \theta_1\} \\
&\quad\leq\quad \mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim D}\{|f(\bar{\mathbf{x}}) - g(\bar{\mathbf{x}})| + |f(\tilde{\mathbf{x}}) - g(\tilde{\mathbf{x}})| \geq \theta_2 - \theta_1\} \\
&\quad\leq\quad \mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim D}\left\{|f(\bar{\mathbf{x}}) - g(\bar{\mathbf{x}})| \geq \frac{\theta_2-\theta_1}{2} \vee |f(\tilde{\mathbf{x}}) - g(\tilde{\mathbf{x}})| \geq \frac{\theta_2-\theta_1}{2}\right\} \\
&\quad\leq\quad \mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim D}\left\{|f(\bar{\mathbf{x}}) - g(\bar{\mathbf{x}})| \geq \frac{\theta_2-\theta_1}{2}\right\} + \mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim D}\left\{|f(\tilde{\mathbf{x}}) - g(\tilde{\mathbf{x}})| \geq \frac{\theta_2-\theta_1}{2}\right\}
\end{aligned}$$

by the union bound. ■

The following lemma is true for every training set S:

Lemma 16 *Let $\mathcal{G}$ be a $\theta/4$-sloppy $\varepsilon/8$-cover for $\mathcal{F}$. Then for all $f \in \mathcal{F}$, there exists $g \in \mathcal{G}$ such that*

$$P_{0,f} - \hat{P}_{\theta,f} \leq P_{\theta/2,g} - \hat{P}_{\theta/2,g} + \frac{\varepsilon}{2}.$$

Proof Let $g \in \mathcal{G}$. Lemma 15, applied to the distribution $\mathcal{D}$, conditioned on $\pi(\bar{\mathbf{x}}, \tilde{\mathbf{x}}) = 1$, implies

$$\mathrm{P}_{0,f} - \mathrm{P}_{\theta/2,g} \leq \mathbb{P}_{\mathbf{x}\sim D_1}\left\{|f(\mathbf{x}) - g(\mathbf{x})| \geq \frac{\theta}{4}\right\} + \mathbb{P}_{\mathbf{x}\sim D_2}\left\{|f(\mathbf{x}) - g(\mathbf{x})| \geq \frac{\theta}{4}\right\}$$

where D_1 and D_2 denote the marginal distributions on $\bar{\mathbf{x}}$ and $\tilde{\mathbf{x}}$, respectively, under distribution $\bar{\mathbf{x}}, \tilde{\mathbf{x}} \sim \mathcal{D}$, conditioned on $\pi(\bar{\mathbf{x}}, \tilde{\mathbf{x}}) = 1$. In other words, for any event $\omega(\mathbf{x})$, $\mathbb{P}_{\mathbf{x}\sim D_1}\{\omega(\mathbf{x})\}$ is the same as $\mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim\mathcal{D}}\{\omega(\bar{\mathbf{x}}) \mid \pi(\bar{\mathbf{x}}, \tilde{\mathbf{x}}) = 1\}$, and similarly $\mathbb{P}_{\mathbf{x}\sim D_2}\{\omega(\mathbf{x})\}$ is the same as $\mathbb{P}_{\bar{\mathbf{x}},\tilde{\mathbf{x}}\sim\mathcal{D}}\{\omega(\tilde{\mathbf{x}}) \mid \pi(\bar{\mathbf{x}}, \tilde{\mathbf{x}}) = 1\}$.

Likewise,

$$\hat{\mathrm{P}}_{\theta/2,g} - \hat{\mathrm{P}}_{\theta,f} \leq \mathbb{P}_{\mathbf{x}\sim S_1}\left\{|f(\mathbf{x}) - g(\mathbf{x})| \geq \frac{\theta}{4}\right\} + \mathbb{P}_{\mathbf{x}\sim S_2}\left\{|f(\mathbf{x}) - g(\mathbf{x})| \geq \frac{\theta}{4}\right\}$$

where S_1 and S_2 are distributions defined analogously for the empirical distribution on S. Thus,

$$\begin{aligned}
\mathrm{P}_{0,f} - \mathrm{P}_{\theta/2,g} + \hat{\mathrm{P}}_{\theta/2,g} - \hat{\mathrm{P}}_{\theta,f} &\leq \mathbb{P}_{\mathbf{x}\sim D_1}\left\{|f(\mathbf{x}) - g(\mathbf{x})| \geq \frac{\theta}{4}\right\} + \mathbb{P}_{\mathbf{x}\sim D_2}\left\{|f(\mathbf{x}) - g(\mathbf{x})| \geq \frac{\theta}{4}\right\} \\
&\quad + \mathbb{P}_{\mathbf{x}\sim S_1}\left\{|f(\mathbf{x}) - g(\mathbf{x})| \geq \frac{\theta}{4}\right\} + \mathbb{P}_{\mathbf{x}\sim S_2}\left\{|f(\mathbf{x}) - g(\mathbf{x})| \geq \frac{\theta}{4}\right\} \\
&= 4\,\mathbb{P}_{\mathbf{x}\sim D^*}\left\{|f(\mathbf{x}) - g(\mathbf{x})| \geq \frac{\theta}{4}\right\} \qquad (18)
\end{aligned}$$

where D^* is the (uniform) mixture of the four distributions D_1, D_2, S_1 and S_2. Constructing D^* in this way allows us to find a g that is close to f for all four terms simultaneously, which is needed for the next step. Since $\mathcal{G}$ is a $\theta/4$-sloppy $\varepsilon/8$-cover, we can now choose g to be a function in the cover $\mathcal{G}$ such that

$$\mathbb{P}_{\mathbf{x}\sim D^*}\left\{|f(\mathbf{x}) - g(\mathbf{x})| \geq \frac{\theta}{4}\right\} \leq \frac{\varepsilon}{8}$$

which, plugging in to equation (18), proves the lemma. ■

In the proof of the theorem, we will use the g's to act as representatives (for slightly different events), so we must show that we do not lose too much by doing this.

Lemma 17 *Let $\mathcal{G}$ be a $\theta/4$-sloppy $\varepsilon/8$-cover for $\mathcal{F}$. Then*

$$\mathbb{P}_{S\sim\mathcal{D}^m}\left\{\exists f \in \mathcal{F} : P_{0,f} - \hat{P}_{\theta,f} \geq \varepsilon\right\} \leq \mathbb{P}_{S\sim\mathcal{D}^m}\left\{\exists g \in \mathcal{G} : P_{\theta/2,g} - \hat{P}_{\theta/2,g} \geq \frac{\varepsilon}{2}\right\}.$$

Proof By Lemma 16, for every training set S, for any $f \in \mathcal{F}$, there exists some $g \in \mathcal{G}$ such that

$$\mathrm{P}_{0,f} - \hat{\mathrm{P}}_{\theta,f} \leq \hat{\mathrm{P}}_{\theta/2,g} - \mathrm{P}_{\theta/2,g} + \frac{\varepsilon}{2}.$$

Thus, if there exists an $f \in \mathcal{F}$ such that $\mathrm{P}_{0,f} - \hat{\mathrm{P}}_{\theta,f} \geq \varepsilon$, then there exists a $g \in \mathcal{G}$ such that $\mathrm{P}_{\theta/2,g} - \hat{\mathrm{P}}_{\theta/2,g} \geq \frac{\varepsilon}{2}$. The statement of the lemma follows directly. ■

Now we incorporate the fact that the training set is chosen randomly. We will use a generalization of Hoeffding's inequality due to McDiarmid, as follows:

Theorem 18 ***(McDiarmid's Inequality McDiarmid 1989)*** *Let $X_1, X_2, \ldots X_m$ be independent random variables under distribution D. Let $f(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ be any real-valued function such that for all $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m; \mathbf{x}_i'$,*

$$|f(\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_m) - f(\mathbf{x}_1, \ldots, \mathbf{x}_i', \ldots \mathbf{x}_m)| \leq c_i.$$

Then for any $\varepsilon > 0$,

$$\begin{aligned}
\mathbb{P}_{X_1,X_2,\ldots,X_m\sim D}\left\{f(X_1, X_2, \ldots, X_m) - \mathbb{E}[f(X_1, X_2, \ldots, X_m)] \geq \varepsilon\right\} &\leq \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^m c_i^2}\right), \\
\mathbb{P}_{X_1,X_2,\ldots,X_m\sim D}\left\{\mathbb{E}[f(X_1, X_2, \ldots, X_m)] - f(X_1, X_2, \ldots, X_m) \geq \varepsilon\right\} &\leq \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^m c_i^2}\right).
\end{aligned}$$

Lemma 19 *For any $f \in \mathcal{F}$,*

$$\mathbb{P}_{S\sim\mathcal{D}^m}\{P_{\theta,f} - \hat{P}_{\theta,f} \geq \varepsilon/2\} \leq 2\exp\left[-\frac{m(\varepsilon E)^2}{8}\right].$$

Proof To make notation easier for this lemma, we introduce some shorthand notation:

$$\begin{aligned}
\mathrm{top}_{\mathcal{D}} &:= \mathbb{E}_{\tilde{\mathbf{x}},\tilde{\mathbf{x}}\sim\mathcal{D}}[\mathbf{1}_{[f(\tilde{\mathbf{x}})-f(\tilde{\mathbf{x}})\leq\theta]}\pi(\tilde{\mathbf{x}},\tilde{\mathbf{x}})] \\
\mathrm{top}_S &:= \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{k=1}^{m}\mathbf{1}_{[f(\mathbf{x}_i)-f(\mathbf{x}_k)\leq\theta]}\pi(\mathbf{x}_i,\mathbf{x}_k) \\
\mathrm{bot}_{\mathcal{D}} := E &:= \mathbb{E}_{\tilde{\mathbf{x}},\tilde{\mathbf{x}}\sim\mathcal{D}}[\pi(\tilde{\mathbf{x}},\tilde{\mathbf{x}})] \\
\mathrm{bot}_S &:= \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{k=1}^{m}\pi(\mathbf{x}_i,\mathbf{x}_k).
\end{aligned}$$

Since diagonal terms have $\pi(\mathbf{x}_i,\mathbf{x}_i)$ which is always 0, $\mathrm{top}_{\mathcal{D}} = \mathbb{E}_{S\sim\mathcal{D}^m}[\mathrm{top}_S]$ and similarly, $\mathrm{bot}_{\mathcal{D}} = \mathbb{E}_{S\sim\mathcal{D}^m}[\mathrm{bot}_S]$. Thus, we can bound the difference between top_S and $\mathrm{top}_{\mathcal{D}}$ using large deviation bounds, and similarly for the difference between bot_S and $\mathrm{bot}_{\mathcal{D}}$. We choose McDiarmid's Inequality to perform this task. It is not difficult to show using the rules of π that the largest possible change in top_S due to the replacement of one example is $1/m$. Similarly the largest possible change in bot_S is $1/m$. Thus, McDiarmid's inequality applied to top_S and bot_S implies that for every $\varepsilon_1 > 0$:

$$\begin{aligned}
\mathbb{P}_{S\sim\mathcal{D}^m}\{\mathrm{top}_{\mathcal{D}} - \mathrm{top}_S \geq \varepsilon_1\} &\leq \exp[-2\varepsilon_1^2 m] \\
\mathbb{P}_{S\sim\mathcal{D}^m}\{\mathrm{bot}_S - \mathrm{bot}_{\mathcal{D}} \geq \varepsilon_1\} &\leq \exp[-2\varepsilon_1^2 m].
\end{aligned}$$

Here, we use ε_1 to avoid confusion with the ε in the statement of the lemma; we will specify ε_1 in terms of ε later, but since the equations are true for any $\varepsilon_1 > 0$, we work with general ε_1 for now. Consider the following event:

$$\mathrm{top}_{\mathcal{D}} - \mathrm{top}_S < \varepsilon_1 \quad \text{and} \quad \mathrm{bot}_S - \mathrm{bot}_{\mathcal{D}} < \varepsilon_1.$$

By the union bound, this event is true with probability at least $1 - 2\exp[-2\varepsilon_1^2 m]$. When the event is true, we can rearrange the equations to be a bound on

$$\frac{\mathrm{top}_{\mathcal{D}}}{\mathrm{bot}_{\mathcal{D}}} - \frac{\mathrm{top}_S}{\mathrm{bot}_S}.$$

We do this as follows:

$$\frac{\mathrm{top}_{\mathcal{D}}}{\mathrm{bot}_{\mathcal{D}}} - \frac{\mathrm{top}_S}{\mathrm{bot}_S} < \frac{\mathrm{top}_{\mathcal{D}}}{\mathrm{bot}_{\mathcal{D}}} - \frac{\mathrm{top}_{\mathcal{D}} - \varepsilon_1}{\mathrm{bot}_{\mathcal{D}} + \varepsilon_1}. \tag{19}$$

If we now choose:

$$\varepsilon_1 = \frac{\varepsilon \mathrm{bot}_{\mathcal{D}}}{2 - \varepsilon + 2\frac{\mathrm{top}_{\mathcal{D}}}{\mathrm{bot}_{\mathcal{D}}}} \geq \frac{\varepsilon \mathrm{bot}_{\mathcal{D}}}{4} =: \frac{\varepsilon E}{4}$$

then the right hand side of (19) is equal to $\varepsilon/2$. Here, we have used $E := \mathrm{bot}_{\mathcal{D}}$, and by the definition of $\mathrm{top}_{\mathcal{D}}$ and $\mathrm{bot}_{\mathcal{D}}$, we always have $\mathrm{top}_{\mathcal{D}} \leq \mathrm{bot}_{\mathcal{D}}$. We directly have:

$$1 - 2\exp[-2\varepsilon_1^2 m] \geq 1 - 2\exp\left(-2m\left[\frac{\varepsilon E}{4}\right]^2\right).$$

Therefore, from our earlier application of McDiarmid, we find that with probability at least

$$1 - 2\exp\left[-\frac{m(\varepsilon E)^2}{8}\right]$$

the following holds:

$$\mathrm{P}_{\theta,f} - \hat{\mathrm{P}}_{\theta,f} = \frac{\mathrm{top}_{\mathcal{D}}}{\mathrm{bot}_{\mathcal{D}}} - \frac{\mathrm{top}_S}{\mathrm{bot}_S} < \varepsilon/2.$$

■

As mentioned earlier, we could have equally well have written the lemma in terms of the empirical quantity $\mathrm{bot}_{\mathcal{D}}$ rather than in terms of E. We have made this decision because the bound is useful for allowing us to determine which quantities are important to maximize in our algorithms; we cannot maximize $\mathrm{bot}_{\mathcal{D}}$ in practice because we are choosing m random instances from $\mathcal{D}$, thus we have no influence at all over the value of $\mathrm{bot}_{\mathcal{D}}$ in practice. Either way, the bound tells us that the margin should be an important quantity to consider in the design of algorithms.

Also, note that this proof implicitly used our simplifying assumption that the truth function π is deterministic. In the more general case, where the value $\pi(\mathbf{x}_i, \mathbf{x}_k)$ of each training pair $\mathbf{x}_i, \mathbf{x}_k$ is determined probabilistically, an alternative proof giving the same result can be given using Azuma's lemma.

Proof *(of Theorem 2)* Let $\mathcal{G}$ be a $\theta/4$-sloppy $\varepsilon/8$-cover of $\mathcal{F}$ of minimum size. Applying Lemma 17, the union bound, and then Lemma 19 for $\theta/2$, we find:

$$\begin{aligned}
\mathbb{P}_{S\sim\mathcal{D}^m}\left\{\exists f \in \mathcal{F} : \mathrm{P}_{0,f} - \hat{\mathrm{P}}_{\theta,f} \geq \varepsilon\right\} &\leq \mathbb{P}_{S\sim\mathcal{D}^m}\left\{\exists g \in \mathcal{G} : \mathrm{P}_{\theta/2,g} - \hat{\mathrm{P}}_{\theta/2,g} \geq \frac{\varepsilon}{2}\right\}. \\
&\leq \sum_{g\in\mathcal{G}} \mathbb{P}_{S\sim\mathcal{D}^m}\left\{\mathrm{P}_{\theta/2,g} - \hat{\mathrm{P}}_{\theta/2,g} \geq \frac{\varepsilon}{2}\right\} \\
&\leq \sum_{g\in\mathcal{G}} 2\exp\left(-\frac{m(\varepsilon E)^2}{8}\right) \\
&= \mathcal{N}\left(\mathcal{F}, \frac{\theta}{4}, \frac{\varepsilon}{8}\right) 2\exp\left[-\frac{m(\varepsilon E)^2}{8}\right].
\end{aligned}$$

Now we put everything together. With probability at least

$$1 - \mathcal{N}\left(\mathcal{F}, \frac{\theta}{4}, \frac{\varepsilon}{8}\right) 2\exp\left[-\frac{m(\varepsilon E)^2}{8}\right],$$

we have

$$\mathbb{P}_D\{\mathrm{misrank}_f\} = \mathrm{P}_{0,f} \leq \hat{\mathrm{P}}_{\theta,f} + \varepsilon = \mathbb{P}_S\{\mathrm{margin}_f \leq \theta\} + \varepsilon.$$

Thus, the theorem has been proved. ■

We now provide a proof for Lemma 3, which gives an estimate of the covering number for convex combinations of dictionary elements.

Proof *(of Lemma 3)* We are trying to estimate the covering number for $\mathcal{F}$, where

$$\mathcal{F} = \left\{ f : f = \sum_j \lambda_j h_j, \sum_j \lambda_j = 1, \forall j\, \lambda_j \geq 0, h_j : \mathcal{X} \to \{0,1\}, h_j \in \mathcal{H} \right\}.$$

Consider the following set $\mathcal{G}_N$ of all g that can be written as a simple average of N elements of $\mathcal{H}$:

$$\mathcal{G}_N = \left\{ \frac{1}{N}(g_1 + \cdots + g_N) : g_1, \ldots, g_N \in \mathcal{H} \right\}.$$

We claim that $\mathcal{G}_N$ is a θ-sloppy ε-cover when

$$N \geq \frac{\ln(2/\varepsilon)}{2\theta^2}. \tag{20}$$

To show this, let f be any function $\mathcal{F}$, and let D be any distribution. We know that $f = \sum_j \lambda_j h_j$ for some λ_j's as above, where the number of terms in the sum can be much more than N. Let us pick N dictionary elements $g_1, \ldots, g_N$ from $\mathcal{H}$ by choosing them randomly and independently with replacement according to the distribution imposed by the coefficients $\boldsymbol{\lambda}$. That is, each g_i is selected to be h_j with probability equal to λ_j. Thus, if λ_j is large, it is more likely that h_j will be chosen as one of the N chosen elements $g_1, \ldots, g_N$. Construct g as the average of those N elements.

Let $\mathbf{x} \in \mathcal{X}$ be any fixed element. Then $g(\mathbf{x})$ is an average of N Bernoulli random variables, namely, $g_1(\mathbf{x}), \ldots, g_N(\mathbf{x})$; by the manner in which each g_j was chosen, each of these Bernoulli random variables is 1 with probability exactly $f(\mathbf{x})$. Therefore, by Hoeffding's inequality,

$$\mathbb{P}_g \left\{ |g(\mathbf{x}) - f(\mathbf{x})| \geq \theta \right\} \leq 2e^{-2\theta^2 N}$$

where $\mathbb{P}_g\{\cdot\}$ denotes probability with respect to the random choice of g.

This holds for every $\mathbf{x}$. Now let $\mathbf{x}$ be random according to D. Then

$$\begin{aligned} \mathbb{E}_g \left[\mathbb{P}_{\mathbf{x}\sim D} \left\{ |f(\mathbf{x}) - g(\mathbf{x})| \geq \theta \right\} \right] &= \mathbb{E}_{\mathbf{x}\sim D} \left[\mathbb{P}_g \left\{ |f(\mathbf{x}) - g(\mathbf{x})| \geq \theta \right\} \right] \\ &\leq \mathbb{E}_{\mathbf{x}\sim D} \left[2e^{-2\theta^2 N} \right] = 2e^{-2\theta^2 N}. \end{aligned}$$

Thus, there exists $g \in \mathcal{G}_N$ such that

$$\mathbb{P}_{\mathbf{x}\sim D} \left\{ |f(\mathbf{x}) - g(\mathbf{x})| \geq \theta \right\} \leq 2e^{-2\theta^2 N}.$$

Hence, selecting N as in equation (20) ensures that $\mathcal{G}_N$ is a θ-sloppy ε-cover. The covering number $\mathcal{N}(\mathcal{F}, \theta, \varepsilon)$ is thus at most

$$|\mathcal{G}_N| \leq |\mathcal{H}|^N,$$

which is the bound given in the statement of the lemma. ■

7. Proofs from Section 4.2

Proof *(of Lemma 9)* There are two possibilities; either $\lim_{t\to\infty} s_t = \infty$ or $\lim_{t\to\infty} s_t < \infty$. We handle these cases separately, starting with the case $\lim_{t\to\infty} s_t = \infty$. From (9),

$$\begin{aligned} s_{t+1}(g_{t+1} - g_t) &= -g_t\alpha_t - \ln\tau_t \\ s_t(g_{t+1} - g_t) &= -g_t\alpha_t - \alpha_t(g_{t+1} - g_t) - \ln\tau_t \\ s_t(g_{t+1} - g_t) &= -\alpha_t g_{t+1} - \ln\tau_t \\ s_t(g_{t+1} - g_t) + \ln\tau_t + \alpha_t &= \alpha_t(1 - g_{t+1}) \geq \alpha_t(1-\rho) \\ \frac{g_{t+1} - g_t}{1-\rho} + \frac{\ln\tau_t + \alpha_t}{s_t(1-\rho)} &\geq \frac{\alpha_t}{s_t} \geq \frac{\alpha_t}{s_{t+1}}. \end{aligned}$$

Since the g_t's constitute a nondecreasing sequence bounded by 1, $(g_{t+1} - g_t) \to 0$ as $t \to \infty$, so the first term on the left vanishes. The second term will vanish as long as we can bound $\ln \tau_t + \alpha_t$ by a constant, since by assumption, $s_t \to \infty$.

We define $g_{\tilde{1}}$ as the first positive value of $\tilde{G}(\boldsymbol{\lambda}_t)$; the value of $\tilde{G}$ only increases from this value. In order to bound $\ln \tau_t + \alpha_t$, we use Equation (11):

$$\begin{aligned} \ln \tau_t + \alpha_t &= \ln(-\tau_t') - \ln g_t + \alpha_t = \ln[d_{t+} e^{-\alpha_t} - d_{t-} e^{\alpha_t}] - \ln g_t + \alpha_t \\ &= \ln[d_{t+} - d_{t-} e^{2\alpha_t}] + \ln e^{-\alpha_t} - \ln g_t + \alpha_t \\ &\leq \ln d_{t+} - \ln g_t \leq \ln 1 - \ln g_{\tilde{1}} = -\ln g_{\tilde{1}} < \infty. \end{aligned}$$

Thus, the second term will vanish, and we now have the sequence α_t / s_{t+1} upper bounded by a vanishing sequence; thus, it too will vanish.

Now for the case where $\lim_{t\to\infty} s_t < \infty$. Consider

$$\begin{aligned} \sum_{t=\tilde{1}}^{T} \frac{\alpha_t}{s_{t+1}} &= \sum_{t=\tilde{1}}^{T} \frac{s_{t+1} - s_t}{s_{t+1}} = \sum_{t=\tilde{1}}^{T} \int_{s_t}^{s_{t+1}} \frac{1}{s_{t+1}} du \\ &\leq \sum_{t=\tilde{1}}^{T} \int_{s_t}^{s_{t+1}} \frac{1}{u} du = \int_{s_{\tilde{1}}}^{s_{T+1}} \frac{1}{u} du = \ln \frac{s_{T+1}}{s_{\tilde{1}}}. \end{aligned}$$

By our assumption that $\lim_{t\to\infty} s_t < \infty$, the above sequence is a bounded increasing sequence. Thus, $\sum_{t=\tilde{1}}^{\infty} \frac{\alpha_t}{s_{t+1}}$ converges. In particular,

$$\lim_{t\to\infty} \frac{\alpha_t}{s_{t+1}} = 0.$$

■

Proof *(of Theorem 7)* The proof relies completely on an important calculus lemma, Lemma 20 below. Before we state the lemma, we make some definitions and derive some tools for later use.

We will be speaking only of iterations t and $t+1$, so when the iteration subscript has been eliminated, it refers to iteration t rather than iteration $t+1$. From now on, the basic independent variables will be r, g and d_0. Here, the ranges are $0 < r < 1$, $0 \leq g < r$, $0 \leq d_0 < \frac{2}{3}(1-r)(1-r^2)$. We change our notation to reinforce this: d_+ and d_- can be considered functions of the basic variables r and d_0 since $d_+ = (1 + r - d_0)/2$ and $d_- = (1 - r - d_0)/2$. Also define $\tau(r, g, d_0) := \tau_t$, $\tau'(r, g, d_0) = \tau_t'$, and $\alpha(r, g, d_0) := \alpha_t$, which are specified by (7), (8) and (11).

Define the following:

$$\begin{aligned} \Gamma(r, g, d_0) &:= \frac{-\ln \tau(r, g, d_0)}{\alpha(r, g, d_0)}. \\ \mathcal{B}(r, g, d_0) &:= \frac{\Gamma(r, g, d_0) - g}{r - g}. \end{aligned}$$

Now we state the important lemma we need for proving the theorem.

Lemma 20 *For* $0 < r < 1$, $0 \leq g < r$, $0 \leq d_0 < \frac{2}{3}(1-r)(1-r^2)$,

$$\mathcal{B}(r, g, d_0) > 1/2.$$

The proof is technical and has been placed in the Appendix. Using only this lemma, we can prove the theorem directly. Let us unravel the notation a bit. From the definition of $\Gamma(r,g,d_0)$ and Lemma 20:

$$\begin{aligned}\frac{-\ln\tau(r,g,d_0)}{\alpha(r,g,d_0)} = \Gamma(r,g,d_0) = g+(r-g)\mathcal{B}(r,g,d_0) &> \frac{r+g}{2}\\ -\ln\tau(r,g,d_0) &> \frac{(r+g)\alpha(r,g,d_0)}{2}.\end{aligned}$$

Using this relation at time t and incorporating the recursive equation, Equation (9),

$$g_{t+1}-g_t = \frac{1}{s_{t+1}}\left[-g_t\alpha_t-\ln\tau_t\right] > \frac{\alpha_t}{s_{t+1}}\left[-g_t+\frac{(r_t+g_t)}{2}\right] = \frac{1}{2}\frac{\alpha_t(r_t-g_t)}{s_{t+1}}.$$

■

We have proved the theorem, minus the proof of Lemma 20 which was the key step. Lemma 20 is a challenging calculus problem in three variables. For the sake of intuition, we plot $\mathcal{B}$ as a function of r and g for fixed $d_0 = 0.01$ in Figure 2. The result of Lemma 20 is apparent, namely that $\mathcal{B}$ is lower bounded by $1/2$.

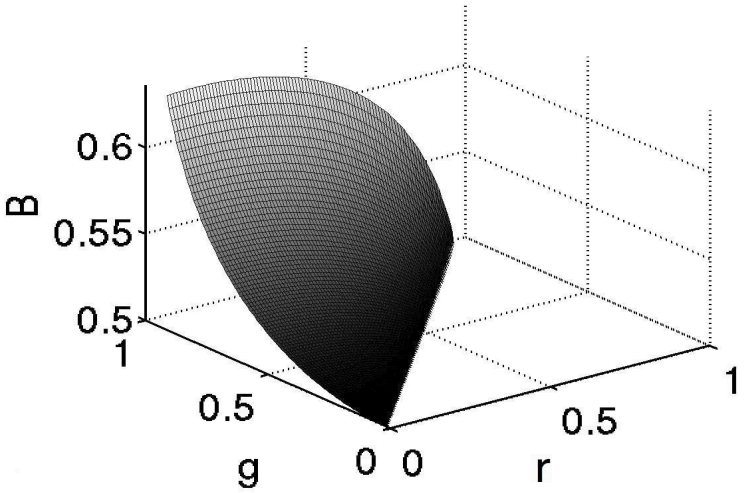


Figure 2: Surface plot of $\mathcal{B}$ as a function of r and g with $d_0 = 0.01$.

8. Proofs from Section 5

Proof *(of Theorem 10)* A proof is only necessary to handle the nonseparable case, since the statement of the theorem is trivial in the separable case. To see this, assume first that we are in the separable case, that is,

$$\lim_{t\to\infty} F_+(\boldsymbol{\lambda}_t) = \lim_{t\to\infty} F_-(\boldsymbol{\lambda}_t) = 0,$$

thus

$$\lim_{t\to\infty} \tilde{F}(\boldsymbol{\lambda}_t) = \lim_{t\to\infty} F(\boldsymbol{\lambda}_t) = 0$$

and we are done. For the rest of the proof, we handle the nonseparable case.

It is possible that the infimum of F or $\tilde{F}$ occurs at infinity, that is, F or $\tilde{F}$ may have no minimizers. Thus, it is not possible to characterize the minimizers by setting the first derivatives to zero. So, in order to more precisely describe the conditions (16) and (17), we now use a technique used by Della Pietra et al. (2002) and later used by Collins et al. (2002), in which we consider F and $\tilde{F}$ as functions of another variable, where the infimum can be achieved. Define, for a particular matrix $\bar{\mathbf{M}}$, the function

$$F_{\bar{\mathbf{M}}}(\boldsymbol{\lambda}) := \sum_{i=1}^{\bar{m}} e^{-(\bar{\mathbf{M}}\boldsymbol{\lambda})_i}.$$

Define

$$\bar{\mathcal{P}} := \{\mathbf{p} | p_i \geq 0 \; \forall i, \; (\mathbf{p}^T \bar{\mathbf{M}})_j = 0 \; \forall \; j\}$$

$$\bar{\mathcal{Q}} := \{\mathbf{q} | q_i = \exp(-(\bar{\mathbf{M}}\boldsymbol{\lambda})_i) \text{ for some } \boldsymbol{\lambda}\}.$$

We may thus consider $\bar{F}_{\bar{\mathbf{M}}}$ as a function of $\bar{\mathbf{q}}$, that is, $\bar{F}_{\bar{\mathbf{M}}}(\bar{\mathbf{q}}) = \sum_{i=1}^{\bar{m}} \bar{q}_i$, where $\bar{\mathbf{q}} \in \bar{\mathcal{Q}}$. We know that since all $\bar{q}_i$'s are positive, the infimum of $\bar{F}$ occurs in a bounded region of $\bar{\mathbf{q}}$ space, which is just what we need.

Theorem 1 of Collins et al. (2002), which is taken directly from Della Pietra et al. (2002), implies that the following are equivalent:

1. $\bar{\mathbf{q}}^* \in \bar{\mathcal{P}} \cap$ closure $(\bar{\mathcal{Q}})$.
2. $\bar{\mathbf{q}}^* \in \text{argmin}_{\bar{\mathbf{q}} \in \text{ closure}(\bar{\mathcal{Q}})} \bar{F}_{\bar{\mathbf{M}}}(\bar{\mathbf{q}})$.

Moreover, either condition is satisfied by exactly one vector $\bar{\mathbf{q}}^*$.

The objective function for AdaBoost is $F = \bar{F}_{\mathbf{M}^{Ada}}$ and the objective for RankBoost is $\tilde{F} = \bar{F}_{\mathbf{M}}$, so the theorem holds for both objectives separately. For the function F, denote $\bar{\mathbf{q}}^*$ as $\mathbf{q}^*$, also $\bar{\mathcal{P}}$ as $\mathcal{P}^{Ada}$ and $\bar{\mathcal{Q}}$ as $\mathcal{Q}^{Ada}$. For the function $\tilde{F}$, denote $\bar{\mathbf{q}}^*$ as $\tilde{\mathbf{q}}^*$, also $\bar{\mathcal{P}}$ as $\tilde{\mathcal{P}}$ and $\bar{\mathcal{Q}}$ as $\tilde{\mathcal{Q}}$. The condition $\mathbf{q}^* \in \mathcal{P}^{Ada}$ can be rewritten as:

$$\sum_{i \in Y_+} q_i^* M_{ij}^{Ada} + \sum_{k \in Y_-} q_k^* M_{kj}^{Ada} = 0 \; \forall \; j. \tag{21}$$

Define $\mathbf{q}_t$ element-wise by: $q_{t,i} := e^{-(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t)_i}$, where the $\boldsymbol{\lambda}_t$'s are a sequence that obey (16), for example, a sequence produced by AdaBoost. Thus, $\mathbf{q}_t \in \mathcal{Q}^{Ada}$ automatically. By assumption, $F(\mathbf{q}_t)$ converges to the minimum of F. Thus, since F is continuous, any limit point of the $\mathbf{q}_t$'s must minimize F as well. But because $\mathbf{q}^*$ is the unique minimizer of F, this implies that $\mathbf{q}^*$ is the one and only ℓ_p-limit point of the $\mathbf{q}_t$'s, and therefore, that the entire sequence of $\mathbf{q}_t$'s converges to $\mathbf{q}^*$ in ℓ_p.

Now define vectors $\tilde{\mathbf{q}}_t$ element-wise by

$$\tilde{q}_{t,ik} := q_{t,i} q_{t,k} = \exp[-(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t)_i - (\mathbf{M}^{Ada}\boldsymbol{\lambda}_t)_k] = \exp[-(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}].$$

Automatically, $\tilde{\mathbf{q}}_t \in \tilde{\mathcal{Q}}$. For any pair i,k the limit of the sequence $\tilde{q}_{t,ik}$ is $\tilde{q}_{ik}^{\infty} := q_i^* q_k^*$.

What we need to show is that $\tilde{\mathbf{q}}^{\infty} = \tilde{\mathbf{q}}^*$. If we can prove this, we will have shown that $\{\boldsymbol{\lambda}_t\}_t$ converges to the minimum of RankBoost's objective function, $\tilde{F}$. We will do this by showing

that $\tilde{\mathbf{q}}^{\infty} \in \tilde{\mathcal{P}}$; once we accomplish this, due to the uniqueness of $\tilde{\mathbf{q}}^*$ as the intersection of $\tilde{\mathcal{P}}$ and closure$(\tilde{Q})$, we will have proved that $\tilde{\mathbf{q}}^{\infty} = \tilde{\mathbf{q}}^*$. So, now we proceed to show $\tilde{\mathbf{q}}^{\infty} \in \tilde{\mathcal{P}}$, using our assumption that the F-skew vanishes. Our assumption that the F-skew vanishes can be rewritten as:

$$\lim_{t\to\infty} \left[\sum_{i\in Y_+} q_{t,i} - \sum_{k\in Y_-} q_{t,k} \right] = 0,$$

that is, since all terms are bounded,

$$\sum_{i\in Y_+} q_i^* = \sum_{k\in Y_-} q_k^*. \tag{22}$$

Consider the quantities $(\tilde{\mathbf{q}}^{\infty T}\mathbf{M})_j$. Remember, if these quantities are zero for every j, then $\tilde{\mathbf{q}}^{\infty} \in \tilde{\mathcal{P}}$ and we have proved the theorem.

$$\begin{aligned}(\tilde{\mathbf{q}}^{\infty T}\mathbf{M})_j &= \sum_{i\in Y_+}\sum_{k\in Y_-} q_i^* q_k^* (M_{ij}^{Ada} + M_{kj}^{Ada}) \\ &= \left(\sum_{k\in Y_-} q_k^*\right)\left(\sum_{i\in Y_+} q_i^* M_{ij}^{Ada}\right) + \left(\sum_{i\in Y_+} q_i^*\right)\left(\sum_{k\in Y_-} q_k^* M_{kj}^{Ada}\right). \end{aligned} \tag{23}$$

Incorporating (22), which is the condition that F-skew$(\mathbf{q}^*) = 0$, (23) becomes:

$$(\tilde{\mathbf{q}}^{\infty T}\mathbf{M})_j = \left(\sum_{i\in Y_+} q_i^*\right)\left[\sum_{i\in Y_+} q_i^* M_{ij}^{Ada} + \sum_{k\in Y_-} q_k^* M_{kj}^{Ada}\right].$$

In fact, according to (21), the bracket in this expression is zero for all j. Thus, $\tilde{\mathbf{q}}_{\infty} \in \tilde{\mathcal{P}}$. We have proved the forward direction of the theorem. The backwards direction, as noted earlier, follows from $(F_+ + F_-)^2 - (F_+ - F_-)^2 = 4F_+F_-$. ■

Proof *(of Corollary 11)* Recall that $\mathbf{q}^* \in \mathcal{P}^{Ada}$. Specifically writing this condition just for the constant weak classifier yields:

$$\begin{aligned} 0 &= \sum_{i\in Y_+} q_i^* M_{i0}^{Ada} + \sum_{k\in Y_-} q_k^* M_{k0}^{Ada} = \sum_{i\in Y_+} q_i^* y_i + \sum_{k\in Y_-} q_k^* y_k \\ &= \sum_{i\in Y_+} q_i^* - \sum_{k\in Y_-} q_k^* = \lim_{t\to\infty} \text{F-skew}(\boldsymbol{\lambda}_t). \end{aligned}$$

■

Proof *(of Theorem 12)* We know from the proof of Theorem 10 that since $\{\boldsymbol{\lambda}_t\}_t$ and $\{\boldsymbol{\lambda}'_t\}_t$ minimize $\tilde{F}$, we automatically have $\tilde{q}_t \to \tilde{q}^*$ and $q'_t \to \tilde{q}^*$ in ℓ_p where

$$q'_{t,ik} := e^{-(\mathbf{M}\boldsymbol{\lambda}'_t)_{ik}}.$$

Thus, we have that for all crucial pairs i,k such that $i \in Y_+$ and $k \in Y_-$:

$$\lim_{t\to\infty} e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}} = \lim_{t\to\infty} e^{-(\mathbf{M}\boldsymbol{\lambda}'_t)_{ik}} = \tilde{q}^*_{ik}.$$

For each crucial pair i,k, if $\tilde{q}^*_{ik} > 1$ then $\lim_{t\to\infty} (\mathbf{M}\boldsymbol{\lambda}_t)_{ik} < 0$, that is,

$$\lim_{t\to\infty} \mathbf{1}_{[(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}\leq 0]} = 1,$$

and conversely, if $\tilde{q}^*_{ik} < 1$ then

$$\lim_{t\to\infty} \mathbf{1}_{[(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}\leq 0]} = 0.$$

This is provided by the continuity of the function $\mathbf{1}_{[z\leq 0]}$ away from $z = 0$, and since there are no asymptotic ties in score as we have assumed, $\tilde{q}^*_{ik} \neq 1$. The same statement holds for $\boldsymbol{\lambda}'_t$. Summing over i,k pairs yields:

$$\lim_{t\to\infty} \left[\sum_{i\in Y_+} \sum_{k\in Y_-} \mathbf{1}_{[(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}\leq 0]} \right] = \lim_{t\to\infty} \left[\sum_{i\in Y_+} \sum_{k\in Y_-} \mathbf{1}_{[(\mathbf{M}\boldsymbol{\lambda}'_t)_{ik}\leq 0]} \right].$$

The theorem has been proved. Note that the AUC value is obtained from this sum by the formula (13). ■

Proof *(of Corollary 13)* By Theorem 10, it is sufficient to show that the correction does not influence the value of $\tilde{F}(\boldsymbol{\lambda}_t)$ and that it makes the F-skew vanish. Consider the vector $\boldsymbol{\lambda}_t + c\mathbf{e}_{j_0}$.

$$\begin{aligned}
\tilde{F}(\boldsymbol{\lambda} + c\mathbf{e}_{j_0}) &= \sum_{i\in Y_+} \sum_{k\in Y_-} \exp\left[-\sum_j \lambda_j h_j(\mathbf{x}_i) - c \right] \exp\left[+\sum_j \lambda_j h_j(\mathbf{x}_k) + c \right] \\
&= \sum_{i\in Y_+} \sum_{k\in Y_-} \exp\left[-\sum_j \lambda_j h_j(\mathbf{x}_i) \right] \exp\left[+\sum_j \lambda_j h_j(\mathbf{x}_k) \right] = \tilde{F}(\boldsymbol{\lambda}).
\end{aligned}$$

So, changing the coefficient of the constant weak classifier will not affect the values of $\tilde{F}(\boldsymbol{\lambda})$. Now, let's compute the F-skew of the corrected sequence:

$$\begin{aligned}
\text{F-skew}(\boldsymbol{\lambda}_t^{\text{corrected}}) &= F_+(\boldsymbol{\lambda}_t + b_t\mathbf{e}_{j_0}) - F_-(\boldsymbol{\lambda}_t + b_t\mathbf{e}_{j_0}) \\
&= \sum_{i\in Y_+} e^{-(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t)_i - b_t} - \sum_{k\in Y_-} e^{-(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t)_k + b_t} \\
&= e^{-b_t} F_+(\boldsymbol{\lambda}_t) - e^{b_t} F_-(\boldsymbol{\lambda}_t) = 0
\end{aligned}$$

where this latter expression is equal to zero by our choice of b_t. Since the F-skew of the corrected sequence is always 0, the corrected sequence will minimize not only RankBoost's objective, but also AdaBoost's. ■

Proof *(of Theorem 14)* We will use a similar argument as in Theorem 12 for misclassification error rather than for ranking error. By assumption, $\boldsymbol{\lambda}_t$ is a sequence that minimizes RankBoost's objective $\tilde{F}$ and the correction forces the F-skew to be zero. Thus $\boldsymbol{\lambda}_t^{\text{corrected}}$ minimizes RankBoost's objective, and Theorem 10 implies that $\boldsymbol{\lambda}_t^{\text{corrected}}$ is also a minimizing sequence for AdaBoost's objective F. Using the same argument as in Theorem 12 substituting AdaBoost for RankBoost, we have that

$$\lim_{t\to\infty} e^{-(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t^{\text{corrected}})_i} =: q_i^*$$

exists for all i and

$$\lim_{t\to\infty} e^{-(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t^{\text{corrected}})_k} =: q_k^*$$

exists for all k. Now, we have that for each example i, if $q_i^* > 1$ then $\lim_{t\to\infty}(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t)_i < 0$, that is,

$$\lim_{t\to\infty} \mathbf{1}_{[(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t^{\text{corrected}})_i \le 0]} = 1,$$

and conversely, if $\tilde{q}_i^* < 1$ then

$$\lim_{t\to\infty} \mathbf{1}_{[(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t^{\text{corrected}})_i \le 0]} = 0.$$

The same holds for all k and for $\boldsymbol{\lambda}_t'^{\text{corrected}}$. Again, there is no asymptotic convergence to the decision boundary as we have assumed, $\tilde{q}_i^* \neq 1$, $\tilde{q}_k^* \neq 1$. The same statement holds for $\boldsymbol{\lambda}_t'^{\text{corrected}}$. Summing over i and k yields:

$$\begin{aligned}
&\lim_{t\to\infty}\left[\sum_{i\in Y_+} \mathbf{1}_{[(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t^{\text{corrected}})_i \le 0]} + \sum_{k\in Y_-} \mathbf{1}_{[(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t^{\text{corrected}})_k \le 0]}\right] \\
&= \lim_{t\to\infty}\left[\sum_{i\in Y_+} \mathbf{1}_{[(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t'^{\text{corrected}})_i \le 0]} + \sum_{k\in Y_-} \mathbf{1}_{[(\mathbf{M}^{Ada}\boldsymbol{\lambda}_t'^{\text{corrected}})_k \le 0]}\right].
\end{aligned}$$

■

9. Conclusions

We have presented three main results. First, in Section 3, we presented a generalization bound for ranking. This bound incorporates a margin, allowing it to be useful in the separable case. The second main result is an algorithm, smooth margin ranking, that maximizes the ranking margin. Our third result is that under very general conditions, AdaBoost solves classification and ranking problems simultaneously, performing just as well for the ranking problem as RankBoost. Conversely, RankBoost with a change in intercept performs just as well for the classification problem as AdaBoost.

10. Open Problems and Future Work

The three main results presented in this paper yield many new directions for future research. We gave a margin-based bound for general ranking. It is worth investigating the design of more specialized margin-based bounds for ranking. We have developed one such bound in Rudin (2009); In that work, we develop a specialized bound based on Theorem 2, designed to emphasize the top portion of the list.

We described a new ranking algorithm, smooth margin ranking, that maximizes the margin. It would be natural to compare the empirical performance of the smooth margin ranking algorithm and RankBoost. In fact, it is also worth considering the empirical performance of AdaBoost to RankBoost, now that we know AdaBoost can be used for ranking.

Acknowledgments

We would like to thank the anonymous reviewers and the editor for their helpful comments; some of these comments were especially helpful in formulating Corollary 13. Thanks to Adrian Banner and Richard Sharp for their patience and assistance with earlier versions of the proof of Lemma 20. Thanks also to Corinna Cortes and Mehryar Mohri, who co-authored a preliminary conference version of this work.

This material is based upon work partially supported by the National Science Foundation under grants IIS-0325500 and CCR-0325463. CDR was supported by an NSF postdoctoral research fellowship under grant DBI-0434636 at New York University.

Appendix A. Proof of Lemma 20

We will first prove some properties of α, τ, Γ, and $\mathcal{B}$ in the following lemmas. First, we show $\alpha(r,g,d_0)$ is a nonnegative, deceasing function of g, and that $\tau(r,g,d_0)$ is an increasing function of g. We also provide a bound on the second derivative of α, which is the key step in the proof of Lemma 20.

Lemma 21 *(Properties of $\alpha(r,g,d_0)$ and $\tau(r,g,d_0)$) For fixed values of r and d_0, considering g as a variable, within the range $0 \le g < r$:*

$$\text{(i)} \quad \lim_{g \to r} \alpha(r,g,d_0) = 0,$$

$$\text{(ii)} \quad \frac{\partial \alpha(r,g,d_0)}{\partial g} = \frac{-\tau(r,g,d_0)}{g\tau'(r,g,d_0) + \tau''(r,g,d_0)} = \frac{-\tau(r,g,d_0)}{(1-g^2)\tau(r,g,d_0) - d_0} < 0,$$

$$\text{(iii)} \quad \lim_{g \to r} \frac{\partial \alpha(r,g,d_0)}{\partial g} = \frac{-1}{1-r^2-d_0} < 0,$$

$$\text{(iv)} \quad \frac{\partial \tau(r,g,d_0)}{\partial g} \ge 0,$$

$$\text{(v)} \quad \tau(r,0,1-r) = 1 - r \le d_0 + \sqrt{(1-d_0)^2 - r^2} = \tau(r,0,d_0),$$

$$\text{(vi)} \quad \frac{\partial^2 \alpha(r,g,d_0)}{\partial g^2} < 0 \text{ whenever } d_0 \le \frac{2}{3}(1-r)(1-r^2) \text{ and } g > 0.$$

Proof By definition

$$\tau(r,g,d_0) = \frac{(1+r-d_0)}{2} e^{-\alpha(r,g,d_0)} + \frac{(1-r-d_0)}{2} e^{\alpha(r,g,d_0)} + d_0,$$

$$\tau'(r,g,d_0) = -\frac{(1+r-d_0)}{2} e^{-\alpha(r,g,d_0)} + \frac{(1-r-d_0)}{2} e^{\alpha(r,g,d_0)},$$

and similarly define $\tau''(r,g,d_0) = \tau(r,g,d_0) - d_0$. Part (i) can be seen from (11), that is, $-\tau'(r,g,d_0) = g\tau(r,g,d_0)$, which simplifies to

$$\frac{(1+r-d_0)}{2} e^{-\alpha(r,g,d_0)} - \frac{(1-r-d_0)}{2} e^{\alpha(r,g,d_0)}$$

$$= g\frac{(1+r-d_0)}{2}e^{-\alpha(r,g,d_0)} + g\frac{(1-r-d_0)}{2}e^{\alpha(r,g,d_0)} + gd_0,$$

so one sets $g = r$ and verifies that $\alpha = 0$ satisfies the equation. Part (ii) is shown by taking implicit derivatives of (11) as follows:

$$\frac{\partial\alpha(r,g,d_0)}{\partial g}(g\tau'(r,g,d_0)+\tau''(r,g,d_0))+\tau(r,g,d_0) = 0,$$

that is,

$$\frac{\partial\alpha(r,g,d_0)}{\partial g} = \frac{-\tau(r,g,d_0)}{g\tau'(r,g,d_0)+\tau''(r,g,d_0)}, \tag{24}$$

and then simplifying using (11) and the definition of $\tau''(r,g,d_0)$. For the inequality, the numerator is negative, and the denominator is (using d_+, d_- notation) $g(-d_+e^{-\alpha}+d_-e^{\alpha})+d_+e^{-\alpha}+d_-e^{\alpha} = (1-g)d_+e^{-\alpha}+(1+g)d_-e^{\alpha} > 0$ since $g < 1$. Part (iii) is shown from (i) and (ii); for $g \to r$, we have $\alpha(r,g,d_0) \to 0$, and thus $\tau(r,g,d_0) \to 1$. The inequality comes from $1-r^2-d_0 > 1-r-d_0 = 2d_- \geq 0$. To show (iv), by the chain rule,

$$\frac{\partial\tau(r,g,d_0)}{\partial g} = \tau'(r,g,d_0)\frac{\partial\alpha(r,g,d_0)}{\partial g}.$$

Since $\tau(r,g,d_0) > 0$ and $\tau'(r,g,d_0) = -g\tau(r,g,d_0)$, we know $\tau'(r,g,d_0) \leq 0$. Additionally, from (ii), $\frac{\partial\alpha}{\partial g} < 0$. Thus (iv) is proved. For (v), we know that when $g = 0$, $\tau'(r,g,d_0) = -g\tau(r,g,d_0)$ means $\tau'(r,0,d_0) = 0$. Using the definition for $\tau'(r,g,d_0)$, we find that $e^{\alpha(r,0,d_0)} = \left(\frac{1+r-d_0}{1-r-d_0}\right)^{1/2}$. Substituting this into the definition of τ yields the equality conditions in (v). The inequality comes from the fact that the right hand side, $d_0+\sqrt{(1-d_0)^2-r^2}$, is monotonically decreasing in d_0. For (vi), a derivative of (24) yields:

$$(g\tau'(r,g,d_0)+\tau''(r,g,d_0))\frac{\partial^2\alpha(r,g,d_0)}{\partial g^2}$$

$$= -\left(\frac{\partial\alpha(r,g,d_0)}{\partial g}\right)\left[\left(\frac{\partial\alpha(r,g,d_0)}{\partial g}\right)(g\tau''(r,g,d_0)+\tau'''(r,g,d_0))+2\tau'(r,g,d_0)\right],$$

where $\tau'''(r,g,d_0) = \tau'(r,g,d_0)$. The left expression (using d_+, d_- notation) is $g\tau'(r,g,d_0)+\tau''(r,g,d_0) = d_+(1-g)e^{-\alpha}+d_-(1+g)e^{\alpha} > 0$ since $g < 1$. Since (ii) shows that $\partial\alpha/\partial g < 0$, we are left to show that the bracketed expression on the right is negative in order for the second derivative of α to be negative. Consider that quantity:

$$\left(\frac{\partial\alpha(r,g,d_0)}{\partial g}\right)(g\tau''(r,g,d_0)+\tau'''(r,g,d_0))+2\tau'(r,g,d_0)$$

$$= \tau'(r,g,d_0)\left[\frac{\partial\alpha(r,g,d_0)}{\partial g}\left(\frac{g\tau''(r,g,d_0)+\tau'(r,g,d_0)}{\tau'(r,g,d_0)}\right)+2\right]$$

and substituting $\tau'(r,g,d_0) = -g\tau(r,g,d_0)$ and $\tau''(r,g,d_0) = \tau(r,g,d_0) - d_0$,

$$\begin{aligned}
&= \tau'(r,g,d_0)\left[\frac{\partial\alpha(r,g,d_0)}{\partial g}\left(\frac{g\tau(r,g,d_0) - gd_0 - g\tau(r,g,d_0)}{-g\tau(r,g,d_0)}\right) + 2\right] \\
&= \tau'(r,g,d_0)\left[\frac{\partial\alpha(r,g,d_0)}{\partial g}\left(\frac{d_0}{\tau(r,g,d_0)}\right) + 2\right] \text{ and from (ii),} \\
&= \tau'(r,g,d_0)\left[\frac{-d_0}{(1-g^2)\tau(r,g,d_0) - d_0} + 2\right]. \qquad (25)
\end{aligned}$$

Since $-\tau'(r,g,d_0) = g\tau(r,g,d_0)$, we know $\tau'(r,g,d_0) < 0$ when $g > 0$. Let us show that the bracketed expression of (25) is positive. Using our assumption on d_0, also $1 - r^2 < 1 - g^2$, (v), and (iv),

$$\begin{aligned}
d_0 &< (1-r^2)(1-r)\frac{2}{3} < (1-g^2)(1-r)\frac{2}{3} = (1-g^2)\tau(r,0,1-r)\frac{2}{3} \\
&\leq (1-g^2)\tau(r,0,d_0)\frac{2}{3} \leq (1-g^2)\tau(r,g,d_0)\frac{2}{3}.
\end{aligned}$$

Rearranging this yields

$$\frac{d_0}{[(1-g^2)\tau(r,g,d_0) - d_0]} < 2.$$

The proof is finished. ■

In order to build up to Lemma 20, we need some properties of $\Gamma(r,g,d_0)$ and B.

Lemma 22 *(Properties of $\Gamma(r,g,d_0)$) For every fixed value of r and d_0, considering g as a variable, within the range $0 \leq g < r$:*

$$\begin{aligned}
&\text{(i)} && \lim_{g\to r}\Gamma(r,g,d_0) = r \\
&\text{(ii)} && \Gamma(r,g,d_0) > g \\
&\text{(iii)} && \frac{\partial\Gamma(r,g,d_0)}{\partial g} > 0 \\
&\text{(iv)} && \Gamma(r,g,d_0) < r.
\end{aligned}$$

Proof The proof of (i) uses L'Hôpital's rule, which we have permission to use from Lemma 21 (i) since $\lim_{g\to r}\alpha(r,g,d_0) = 0$.

$$\lim_{g\to r}\Gamma(r,g,d_0) = \lim_{g\to r}\frac{-\ln\tau(r,g,d_0)}{\alpha(r,g,d_0)} = \lim_{g\to r}\frac{-\frac{\tau'(r,g,d_0)}{\tau(r,g,d_0)}\frac{\partial\alpha(r,g,d_0)}{\partial g}}{\frac{\partial\alpha(r,g,d_0)}{\partial g}} = \lim_{g\to r} g = r.$$

Here we have used that $\lim_{g\to r}\frac{\partial\alpha(r,g,d_0)}{\partial g}$ is finite from Lemma 21 (ii), and applied (11), that is, $-\tau'(r,g,d_0) = g\tau(r,g,d_0)$.

For the proofs of (ii) and (iii) we consider the derivative of $\Gamma(r,g,d_0)$ with respect to g. Recall that $\tau'(r,g,d_0)$ is given by the formula (8).

$$\begin{aligned}
\frac{\partial\Gamma(r,g,d_0)}{\partial g} &= \left[\frac{-\tau'(r,g,d_0)}{\tau(r,g,d_0)} + \frac{\ln\tau(r,g,d_0)}{\alpha(r,g,d_0)}\right]\frac{1}{\alpha(r,g,d_0)}\frac{\partial\alpha(r,g,d_0)}{\partial g} \\
&= (\Gamma(r,g,d_0) - g)\left(-\frac{\partial\alpha(r,g,d_0)}{\partial g}\right)\frac{1}{\alpha(r,g,d_0)}. \qquad (26)
\end{aligned}$$

For the last line above we used (11) and the definition of $\Gamma(r,g,d_0)$. Since $\alpha(r,g,d_0)$ is a positive, decreasing function of g from Lemma 21 (ii), we know $-\partial\alpha(r,g,d_0)/\partial g$ and $1/\alpha(r,g,d_0)$ are positive. Thus,

$$\text{sign}\left(\frac{\partial\Gamma(r,g,d_0)}{\partial g}\right) = \text{sign}(\Gamma(r,g,d_0)-g). \tag{27}$$

We show next that $\Gamma(r,0,d_0) > 0$. From (11), we know $0 = -\tau'(r,0,d_0)$, which by definition of $\tau'(r,g,d_0)$ in (8) gives $\alpha(r,0,d_0) = \frac{1}{2}\ln(d_+/d_-) > 0$. Now,

$$\Gamma(r,0,d_0) = \frac{-\ln\tau(r,0,d_0)}{\alpha(r,0,d_0)} = \frac{1}{\alpha(r,0,d_0)}\left(-\ln\left(2(d_-d_+)^{1/2}+d_0\right)\right).$$

We also have $2(d_-d_+)^{1/2}+d_0 < d_-+d_++d_0 = 1$, so we are done showing that $\Gamma(r,0,d_0) > 0$.

Now we proceed by contradiction. Assume that there is some value of $\bar{g}$, where $0 \le \bar{g} < r$, for which $\Gamma(r,\bar{g},d_0) \le \bar{g}$. That is, assume the functions $\Gamma(r,g,d_0)$ and $f(g) = g$ cross. In that case, the derivative $\partial\Gamma(r,g,d_0)/\partial g$ would have a nonpositive sign at $g = \bar{g}$ by (27), and the function $\Gamma(r,g,d_0)$ would be a nonincreasing function for $\bar{g} < g < r$. That is, since $\Gamma(r,g,d_0)$ would have a nonpositive slope at $\bar{g}$, it cannot increase to cross the line $f(g) = g$ in order to reverse the sign of the slope. However, this is a contradiction, since the function must indeed increase; it must reach the limiting value r as $g \to r$, as we showed in (i). Hence, $\Gamma(r,g,d_0) > g$ for all g such that $0 \le g < r$, proving (ii), and thus by (27), $\partial\Gamma(r,g,d_0)/\partial g > 0$ for all g such that $0 \le g < r$, proving (iii).

The proof of (iv) is again by contradiction. Fix arbitrary values of r and d_0. Assume $\Gamma(r,\bar{g},d_0) \ge r$ for some $\bar{g} < r$. Since the function $\Gamma(r,g,d_0)$ is an increasing function of g, $\Gamma(r,g,d_0)$ must be larger than r and strictly increasing for $g > \bar{g}$. Yet by (i), $\Gamma(r,g,d_0) \to r$ as $g \to r$ for each fixed pair of r and d_0. This is a contradiction, since $\Gamma(r,g,d_0)$ cannot decrease to meet this limit. ■

Lemma 23

(i) $\quad 0 < \mathcal{B}(r,g,d_0) < 1$

(ii) $\quad \lim_{g\to r}\mathcal{B}(r,g,d_0) = \frac{1}{2}$ *for fixed* r *and* d_0.

Proof From Lemma 22 (ii), $\Gamma(r,g,d_0)-g$ is positive, and by assumption $g < r$. Thus, $\mathcal{B}(r,g,d_0) > 0$. Also, from Lemma 22 (iv), $\Gamma(r,g,d_0) < r$. Thus, $\mathcal{B}(r,g,d_0) < 1$. Thus (i) is proved. The proof of (ii) uses L'Hôpital's rule twice (which we may use by Lemma 21 (i)) also (11), and the fact that derivatives of $\alpha(r,g,d_0)$ with respect to g are finite.

$$\begin{aligned}
\lim_{g\to r}\mathcal{B}(r,g,d_0) &= \lim_{g\to r}\frac{\frac{-\ln\tau(r,g,d_0)}{\alpha(r,g,d_0)}-g}{r-g} = \lim_{g\to r}\frac{-\ln\tau(r,g,d_0)-g\alpha(r,g,d_0)}{\alpha(r,g,d_0)(r-g)} \\
&= \lim_{g\to r}\frac{-\frac{\tau'(r,g,d_0)}{\tau(r,g,d_0)}\frac{\partial\alpha(r,g,d_0)}{\partial g}-g\frac{\partial\alpha(r,g,d_0)}{\partial g}-\alpha(r,g,d_0)}{-\alpha(r,g,d_0)+(r-g)\frac{\partial\alpha(r,g,d_0)}{\partial g}} \\
&= \lim_{g\to r}\frac{-\alpha(r,g,d_0)}{-\alpha(r,g,d_0)+(r-g)\frac{\partial\alpha(r,g,d_0)}{\partial g}} \\
&= \lim_{g\to r}\frac{-\frac{\partial\alpha(r,g,d_0)}{\partial g}}{-2\frac{\partial\alpha(r,g,d_0)}{\partial g}+(r-g)\frac{\partial^2\alpha(r,g,d_0)}{\partial g^2}} = \frac{1}{2}.
\end{aligned}$$

■

There is one thing left in order to prove Lemma 20. This is where the key step appears, that is, our bound on the second derivative of α.

Lemma 24

$$\frac{(r-g)}{\alpha(r,g,d_0)}\left(-\frac{\partial\alpha(r,g,d_0)}{\partial g}\right) < 1.$$

Proof Define

$$\phi(r,g,d_0) := (r-g)\left(-\frac{\partial\alpha(r,g,d_0)}{\partial g}\right) - \alpha(r,g,d_0).$$

In order to prove the lemma, we need only to show that $\phi(r,g,d_0)$ is always negative. We will show that $\partial\phi(r,g,d_0)/\partial g$ is positive. Thus, the largest possible value of $\phi(r,g,d_0)$ occurs when g is at its maximum, namely, when $g = r$. If $g = r$, then $\phi(r,g,d_0) = 0$. Thus, $\phi(r,g,d_0)$ is everywhere negative and the lemma is proved. Now we have only to prove $\partial\phi(r,g,d_0)/\partial g$ is positive. Again, we take derivatives:

$$\frac{\partial\phi(r,g,d_0)}{\partial g} = (r-g)\left(-\frac{\partial^2\alpha(r,g,d_0)}{\partial g^2}\right),$$

and since $r-g$ is always positive, and since we have taken efforts to ensure α's second derivative is negative (except at the irrelevant endpoint $g = 0$) in Lemma 21 (vi), we are done. ■

We finally prove Lemma 20.

Proof *(of Lemma 20)* We consider $\partial\mathcal{B}(r,g,d_0)/\partial g$ for each fixed pair of r and d_0 values and derive a differential equation for $\mathcal{B}$. We will prove that the derivative is always nonnegative. Then we will use Lemma 23 to show that $\mathcal{B}(r,g,d_0)$ is nonnegative. Here is the differential equation:

$$\begin{aligned}
\frac{\partial\mathcal{B}(r,g,d_0)}{\partial g} &= \frac{1}{r-g}\left[\frac{\partial\Gamma(r,g,d_0)}{\partial g} - 1 + \frac{\Gamma(r,g,d_0)-g}{r-g}\right] \\
&= \frac{1}{r-g}\left[\frac{\partial\Gamma(r,g,d_0)}{\partial g} - 1 + \mathcal{B}(r,g,d_0)\right] \\
&= \frac{1}{r-g}\left[(\Gamma(r,g,d_0)-g)\left(-\frac{\partial\alpha(r,g,d_0)}{\partial g}\frac{1}{\alpha(r,g,d_0)}\right) - 1 + \mathcal{B}(r,g,d_0)\right] \\
&= \frac{1}{r-g}\left[\mathcal{B}(r,g,d_0)(r-g)\left(-\frac{\partial\alpha(r,g,d_0)}{\partial g}\frac{1}{\alpha(r,g,d_0)}\right) - 1 + \mathcal{B}(r,g,d_0)\right] \\
&= \frac{\mathcal{B}(r,g,d_0)}{r-g}\left[(r-g)\left(-\frac{\partial\alpha(r,g,d_0)}{\partial g}\frac{1}{\alpha(r,g,d_0)}\right) - \left(\frac{1}{\mathcal{B}(r,g,d_0)} - 1\right)\right]. \qquad (28)
\end{aligned}$$

Here we have incorporated the differential equation for $\Gamma(r,g,d_0)$ from (26). Again, we will prove by contradiction. Assume that for some values of r and g, where $g < r$, we have $\mathcal{B}(r,g,d_0) \leq 1/2$. That is, assume $\left(\frac{1}{\mathcal{B}(r,g,d_0)} - 1\right) \geq 1$. In that case, the bracket in Equation (28) is negative, by Lemma 24. Since $0 < \mathcal{B}(r,g,d_0) < 1$ from Lemma 23, and $g < r$ by assumption, the factor $\mathcal{B}(r,g,d_0)/(r-g)$ of Equation (28) is positive and the bracket is negative, thus $\frac{\partial\mathcal{B}(r,g,d_0)}{\partial g} < 0$, so $\mathcal{B}(r,g,d_0)$ is a decreasing function. Hence, for each fixed r and d_0, $\mathcal{B}(r,g,d_0)$ decreases from a value

which is less than or equal to $1/2$. Recall from Lemma 23 that $\lim_{g \to r} \mathcal{B}(r,g,d_0) = 1/2$, and thus this limit can never be attained. Contradiction. Thus, for all values of r, d_0 and g within $0 < r < 1$, $0 \leq g < r, 0 \leq d_0 < \frac{2}{3}(1-r)(1-r^2)$, we must have $\mathcal{B}(r,g,d_0) > 1/2$. We have proved the lemma. ■

References

Shivani Agarwal, Thore Graepel, Ralf Herbich, Sariel Har-Peled, and Dan Roth. Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research*, 6:393–425, 2005.

Peter 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(2):525–536, March 1998.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

Olivier Bousquet. New approaches to statistical learning theory. *Annals of the Institute of Statistical Mathematics*, 55(2):371–389, 2003.

Ulf Brefeld and Tobias Scheffer. AUC maximizing support vector learning. In *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning*, 2005.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 89–96, 2005.

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the Twenty-third International Conference on Machine Learning*, 2006.

Stéphan Clemençon and Nicolas Vayatis. Ranking the best instances. *Journal of Machine Learning Research*, 8:2671–2699, Dec 2007.

Stéphan Clemençon, Gabor Lugosi, and Nicolas Vayatis. Ranking and scoring using empirical risk minimization. In *Proceedings of the Eighteenth Annual Conference on Learning Theory*, 2005.

Stéphan Clemençon, Gabor Lugosi, and Nicolas Vayatis. Ranking and empirical minimization of U-statistics. *The Annals of Statistics*, 36(2):844–874, 2007.

Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1/2/3), 2002.

Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems 16*, 2004.

Corinna Cortes and Mehryar Mohri. Confidence intervals for the area under the ROC curve. In *Advances in Neural Information Processing Systems 17*, 2005.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

David Cossock and Tong Zhang. Statistical analysis of bayes optimal subset ranking. *IEEE Trans. Info. Theory*, 54:4140–5154, 2008.

Ofer Dekel, Christopher Manning, and Yoram Singer. Log-linear models for label ranking. In *Advances in Neural Information Processing Systems 16*, 2004.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Duality and auxiliary functions for Bregman distances. Technical Report CMU-CS-01-109R, School of Computer Science, Carnegie Mellon University, 2002.

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.

Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behaviour*, 29:79–103, 1999.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

Martin Gardner. *The Colossal Book of Mathematics*, chapter on More Nontransitive Paradoxes, pages 297–311. Norton, 2001.

Quoc Le and Alex Smola. Direct optimization of ranking measures. arXiv:0704.3359v1, November 2007.

Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics 1989*, pages 148–188. Cambridge University Press, 1989.

Alain Rakotomamonjy. Optimizing AUC with support vector machine (SVM). In *Proceedings of European Conference on Artificial Intelligence Workshop on ROC Curve and AI, Valencia, Spain*, 2004.

Cynthia Rudin. The P-Norm Push: A simple convex ranking algorithm that concentrates at the top of the list. *Journal of Machine Learning Research*, 10:2233–2271, October 2009.

Cynthia Rudin, Ingrid Daubechies, and Robert E. Schapire. The dynamics of AdaBoost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research*, 5:1557–1595, December 2004.

Cynthia Rudin, Corinna Cortes, Mehryar Mohri, and Robert E. Schapire. Margin-based ranking meets boosting in the middle. In *Proceedings of the Eighteenth Annual Conference on Learning Theory*, pages 63–78, 2005.

Cynthia Rudin, Robert E. Schapire, and Ingrid Daubechies. Analysis of boosting algorithms using the smooth margin function. *The Annals of Statistics*, 35(6):2723–2768, 2007.

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999.

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.

Shai Shalev-Shwartz and Yoram Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 7:1567–1599, December 2006.

Nicolas Usunier, Massih-Reza Amini, and Patrick Gallinari. A data-dependent generalisation error bound for the AUC. In *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning*, 2005.

Tong Zhang and Bin Yu. Boosting with early stopping - convergence and consistency. *The Annals of Statistics*, 33(4):1538–1579, 2005.

Journal of Machine Learning Research 10 (2009) 2233-2271 Submitted 4/08; Revised 10/08; Published 10/09

The P-Norm Push: A Simple Convex Ranking Algorithm that Concentrates at the Top of the List

Cynthia Rudin[*] RUDIN@MIT.EDU
MIT Sloan School of Management
Cambridge, MA 02142

Editor: Michael Collins

Abstract

We are interested in supervised ranking algorithms that perform especially well near the top of the ranked list, and are only required to perform sufficiently well on the rest of the list. In this work, we provide a general form of convex objective that gives high-scoring examples more importance. This "push" near the top of the list can be chosen arbitrarily large or small, based on the preference of the user. We choose ℓ_p-norms to provide a specific type of push; if the user sets p larger, the objective concentrates harder on the top of the list. We derive a generalization bound based on the p-norm objective, working around the natural asymmetry of the problem. We then derive a boosting-style algorithm for the problem of ranking with a push at the top. The usefulness of the algorithm is illustrated through experiments on repository data. We prove that the minimizer of the algorithm's objective is unique in a specific sense. Furthermore, we illustrate how our objective is related to quality measurements for information retrieval.

Keywords: ranking, RankBoost, generalization bounds, ROC, information retrieval

1. Introduction

The problem of supervised ranking is useful in many application domains, for instance, maintenance operations to be performed in a specific order, natural language processing, information retrieval, and drug discovery. Many of these domains require the construction of a ranked list, yet often, only the top portion of the list is used in practice. For instance, in the setting of supervised movie ranking, the learning algorithm provides the user (an avid movie-goer) with a ranked list of movies based on preference data. We expect the user to examine the top portion of the list as a recommendation. It is possible that she never looks at the rest of the list, or examines it only briefly. Thus, we wish to make sure that the top portion of the list is correctly constructed. This is the problem on which we concentrate.

We present a fairly general and flexible technique for solving these types of problems. Specifically, we derive a convex objective function that places more emphasis at the top of the list. The algorithm we develop using this technique ("The P-Norm Push") is based on minimization of a specific version of this objective. The user chooses a parameter "p" in the objective, corresponding to the p of an ℓ_p norm. By varying p, one changes the degree of concentration ("push") at the top of the list. One can concentrate at the very top of the list (a big push, large p), or one can have a moderate emphasis at the top (a little push, low p), or somewhere in between. The case with no

∗. Also at Center for Computational Learning Systems, Columbia University, 475 Riverside Drive MC 7717, New York, NY 10115.

emphasis at the top (no push, $p = 1$) corresponds to a standard objective for supervised bipartite ranking, namely the exponentiated pairwise misranking error.

The P-Norm Push is motivated in the setting of supervised bipartite ranking. In the supervised bipartite ranking problem, each training instance has a label of +1 or -1; each movie is either a good movie or a bad movie. In this case, we want to push the bad movies away from the top of the list where the good movies are desired. The quality of a ranking can be determined by examining the Receiver Operator Characteristic (ROC) curve. The AUC (Area Under the ROC Curve) is precisely a constant times one minus the total standard pairwise misranking error. The accuracy measure for our problem is different; we care mostly about the leftmost portion of the ROC curve, corresponding to the top of the ranked list. We wish to make the leftmost portion of the curve higher. Thus, we choose to make a tradeoff: in order make the leftmost portion of the curve higher, we sacrifice on the total area underneath the curve. The parameter p in the P-Norm Push allows the user to directly control this tradeoff.

This problem is highly asymmetric with respect to the positive and negative classes, and is not represented by a sum of independent random variables. It is interesting to consider generalization bounds for such a problem; it is not clear how to use standard techniques that require natural symmetry with respect to the positive and negative examples, for instance, many VC bounds rely on this kind of symmetry. In this work, we present a generalization bound that uses covering numbers as a measure of complexity. This bound is designed specifically to handle these asymmetric conditions. The bound underscores an important property of algorithms that concentrate on a small portion of the domain, such as algorithms that concentrate on the top of a ranked list: these algorithms require more examples for generalization.

Recently, there has been a large amount of interest in the supervised ranking problem, and especially in the bipartite problem. Freund et al. (2003) have developed the RankBoost algorithm for the general setting. We inherit the setup of RankBoost, and our algorithm will also be a boosting-style algorithm. Oddly, Freund and Schapire's classification algorithm AdaBoost (Freund and Schapire, 1997) performs just as well for bipartite ranking as RankBoost; both algorithms achieve equally good values of the AUC (Rudin and Schapire, 2009). This is in contrast with support vector machine classifiers (Cortes and Vapnik, 1995), which do not tend to perform well for the bipartite ranking problem (Rakotomamonjy, 2004; Brefeld and Scheffer, 2005). Mozer et al. (2002) aim to manipulate specific points of the ROC curve in order to study "churn" in the telecommunications industry. Perhaps the closest algorithm to ours is the one proposed by Dekel et al. (2004), who have used a similar form of objective with different specifics to achieve a different goal, namely to rank labels in a multilabel setting. Other related works on label ranking include those of Crammer and Singer (2001) and Shalev-Shwartz and Singer (2006). The work of Yan et al. (2003) contains a brief mention of a method to optimize the lower left corner of the ROC curve, though their multi-layer perception approach is highly non-convex. There is a lot of recent work on generalization bounds (and large deviation bounds) for supervised ranking, namely, the bounds of Freund et al. (2003), Clemençon et al. (2008), Agarwal et al. (2005), Usunier et al. (2005), Hill et al. (2002), Rudin et al. (2005) and Rudin and Schapire (2009), though we were only able to adapt techniques from the latter two bounds to our particular setting, since the covering number approach can handle the natural asymmetry of our problem. There is also a body of work on ROC curves in general, for example, the estimation of confidence bands for ROC curves (Macskassy et al., 2005), and more recent works by Clemençon and Vayatis addressing statistical aspects of ranking problems (e.g., Clemençon and Vayatis, 2007, 2008).

There is a large body of literature on information retrieval (IR) that considers other quality measurements for a ranked list, including "discounted cumulative gain," "average precision" and "winner take all." In essence, the P-Norm Push algorithm can be considered as a way to interpolate between AUC maximization (no push, $p = 1$) and a quantity similar to "winner take all" (largest possible push, $p = \infty$). A simple variation of the P-Norm Push derivation can be used to derive convex objectives that are somewhat similar to the "discounted cumulative gain" as we illustrate in Section 7. Our approach yields simple smooth convex objectives that can be minimized using simple coordinate techniques. In that sense, our work complements those of Tsochantaridis et al. (2005) and Le and Smola (2007) who also minimize a convex upper bound of IR ranking measurements, but with a structured learning approach that requires optimization with exponentially many constraints; those works have suggested useful ways to combat this problem. Additionally, there are recent works (Cossock and Zhang, 2006; Zheng et al., 2007) that suggest regression approaches to optimize ranking criteria for information retrieval.

Here is the outline of the work: in Section 2, we present a general form of objective function, allowing us to incorporate a push near the top of the ranked list. In order to construct a specific case of this objective, one chooses both a loss function ℓ and a convex "price" function g. We will choose g to be a power law, $g(r) = r^p$, so that a higher power p corresponds to a larger push near the top. In Section 3 we give some examples to illustrate how the objective works. In Section 4, we provide a generalization bound for our objective with ℓ as the 0-1 loss, based on L_∞ covering numbers. The generalization bound has been improved from the conference version of this work (Rudin, 2006). In Section 5 we derive the "P-Norm Push" coordinate descent algorithm based on the objective with ℓ chosen as the exponential loss used for AdaBoost and RankBoost. Section 6 discusses uniqueness of the minimizer of the P-Norm Push algorithm's objective. We prove that the minimizer is unique in a specific sense. This result is based on conjugate duality and the theory of Bregman distances (Della Pietra et al., 2002), and is analogous to the result of Collins et al. (2002) for AdaBoost. The "primal" problem for AdaBoost can be written as relative entropy minimization. For the objective of the P-Norm Push algorithm, the problem is more difficult and the primal is not a common function. Section 7 illustrates the similarity between quality measurements used for information retrieval and our objective, and gives other variations of the objective. In Section 8, we demonstrate the P-Norm Push on repository data. Section 9 discusses open problems and future work. Sections 10 and 11 contain the major proofs from Sections 4 and 6. The P-Norm Push was recently applied to the problem of prioritizing manholes in New York City for maintenance and repair (Rudin et al., 2009).

The main contributions of this work are: a generalization bound for a learning problem that is asymmetric by design, a simple user-adjustable, easy-to-implement algorithm for supervised ranking with a "push," and a proof that the minimizer of the algorithm's objective is unique in a specific sense.

2. An Objective for Ranking with a Push

The set of instances with positive labels is $\{\mathbf{x}_i\}_{i=1,\ldots,I}$, where $\mathbf{x}_i \in \mathcal{X}$. The negative instances are $\{\tilde{\mathbf{x}}_k\}_{k=1,\ldots,K}$, where $\tilde{\mathbf{x}}_k \in \mathcal{X}$. We always use i for the index over positive instances and k for the index over negative instances. In the case of the movie ranking problem, the $\mathbf{x}_i$'s are the good movies used for training, the $\tilde{\mathbf{x}}_k$'s are the bad movies, and $\mathcal{X}$ is a database of movies. Our goal is to construct a ranking function f that gives a real valued score to each instance in $\mathcal{X}$, that is, $f : \mathcal{X} \to \mathcal{R}$. We do

not care about the actual values of each instance, only the relative values; for positive-negative pair $\mathbf{x}_i, \tilde{\mathbf{x}}_k$, we care that $f(\mathbf{x}_i) > f(\tilde{\mathbf{x}}_k)$ but it is not important to know, for example, that $f(\mathbf{x}_i) = 0.4$ and $f(\tilde{\mathbf{x}}_k) = 0.1$.

Let us now derive the general form of our objective. For a particular negative example, we wish to reduce its *Height*, which is the number of positive examples ranked beneath it. That is, for each k, we wish to make Height(k) small, where:

$$\text{Height}(k) := \sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}.$$

Let us now add the push. We want to concentrate harder on negative examples that have high scores; we want to push these examples down from the top. Since the highest scoring negative examples also achieve the largest Heights, these are the examples for which we impose a larger price. Namely, for convex, non-negative, monotonically increasing function $g : \mathcal{R}_+ \to \mathcal{R}_+$, we place the price $g(\text{Height}(k))$ on negative example k:

$$g\left(\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}\right).$$

If g is very steep, we pay an extremely large price for a high scoring negative example. Examples of steep functions include $g(r) = e^r$ and $g(r) = r^p$ for p large. Thus we have derived an objective to minimize, namely the sum of the prices for the negative examples:

$$R_{g,\mathbf{1}}(f) := \sum_{k=1}^{K} g\left(\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}\right).$$

The effect of g is to force the value of $R_{g,\mathbf{1}}$ to come mostly from the highest scoring negative examples. These high scoring negative examples are precisely the examples represented by the leftmost portion of the ROC Curve. Minimizing $R_{g,\mathbf{1}}$ should thus boost performance around high scoring negative examples and increase the leftmost portion of the ROC Curve.

It is hard to minimize $R_{g,\mathbf{1}}$ directly due to the 0-1 loss in the inner sum. Instead, we will minimize an upper bound, $R_{g,\ell}$, which incorporates $\ell : \mathcal{R} \to \mathcal{R}_+$, a convex, non-negative, monotonically decreasing upper bound on the 0-1 loss. Popular loss functions include the exponential, logistic, and hinge losses. We can now define the general form of our objective:

$$R_{g,\ell}(f) := \sum_{k=1}^{K} g\left(\sum_{i=1}^{I} \ell\Big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\Big)\right).$$

To construct a specific version of this objective, one chooses the loss ℓ, the price function g, and an appropriate hypothesis space $\mathcal{F}$ over which to minimize $R_{g,\ell}$. In order to derive RankBoost's specific objective from $R_{g,\ell}$, we would choose ℓ as the exponential loss and g to be the identity.

For the moment, let us assume we care only about the very top of the list, that is, we wish to push the most offending negative example as far down the list as possible. Equivalently, we wish to minimize $R_{\max}$, the number of positives below the highest scoring negative example:

$$R_{\max}(f) := \max_k \sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}.$$

Minimizing this misranking error at the very top is similar to optimizing a "winner take all" loss such as $\mathbf{1}_{[\max_i f(\mathbf{x}_i) \leq \max_k f(\tilde{\mathbf{x}}_k)]}$ in that both would choose a ranked list where a negative example is not at the top of the list.

Although it is hard to minimize $R_{\max}(f)$ directly, $R_{g,\ell}$ can give us some control over $R_{\max}$. Namely, the following relationships exist between $R_{g,\ell}$, $R_{g,\mathbf{1}}$ and $R_{\max}$.

Theorem 1 *For all convex, non-negative, monotonic g and for all ℓ that are upper bounds for the 0-1 loss, we have that:*

$$Kg\left(\frac{1}{K}R_{\max}(f)\right) \leq R_{g,\mathbf{1}}(f) \leq Kg\Big(R_{\max}(f)\Big) \quad \text{and} \quad R_{g,\mathbf{1}}(f) \leq R_{g,\ell}(f).$$

Proof The proof of the first inequality follows from the monotonicity of g and Jensen's inequality for convex function g.

$$\begin{aligned} Kg\left(\frac{1}{K}R_{\max}(f)\right) &= Kg\left(\frac{1}{K}\max_k \sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i)\leq f(\tilde{\mathbf{x}}_k)]}\right) \leq Kg\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i)\leq f(\tilde{\mathbf{x}}_k)]}\right) \\ &\leq \sum_{k=1}^{K} g\left(\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i)\leq f(\tilde{\mathbf{x}}_k)]}\right) = R_{g,\mathbf{1}}(f). \end{aligned}$$

For the second inequality, we use the fact that g is monotonic:

$$\begin{aligned} R_{g,\mathbf{1}}(f) &= \sum_{k=1}^{K} g\left(\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i)\leq f(\tilde{\mathbf{x}}_k)]}\right) \leq K\max_k g\left(\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i)\leq f(\tilde{\mathbf{x}}_k)]}\right) \\ &= Kg\left(\max_k \sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i)\leq f(\tilde{\mathbf{x}}_k)]}\right) = Kg\Big(R_{\max}(f)\Big). \end{aligned}$$

Using that ℓ is an upper bound on the 0-1 loss, we have the last inequality:

$$R_{g,\mathbf{1}}(f) = \sum_{k=1}^{K} g\left(\sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i)\leq f(\tilde{\mathbf{x}}_k)]}\right) \leq \sum_{k=1}^{K} g\left(\sum_{i=1}^{I} \ell\Big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\Big)\right) = R_{g,\ell}(f).$$

■

The fact that the function $Kg(\frac{1}{K}r)$ is monotonic in r adds credibility to our choice of objective $R_{g,\ell}$; if $R_{g,\ell}(f)$ is minimized, causing a reduction in $Kg(\frac{1}{K}R_{\max}(f))$, then $R_{\max}(f)$ will also be reduced. Thus, Theorem 1 suggests that $R_{g,\ell}$ is a reasonable quantity to minimize in order to incorporate a push at the top, for instance, in order to diminish $R_{\max}$. Also recall that if g is especially steep, for instance $g(r) = e^r$ or $g(r) = r^p$ for p large, then $g^{-1}(\sum_{k=1}^{K} g(r_k)) \approx \max_k r_k$. That is, the quantity $g^{-1}(R_{g,\mathbf{1}})$, for steep functions g, will approximate $R_{\max}$.

For most of the paper, we are considering the power law (or "p-norm") price functions $g(r) = r^p$. By allowing the user to choose p, we allow the amount of push to be specified to match the application. At the heart of this derivation, we are using ℓ_p-norms to interpolate between the ℓ_1-norm (the AUC), and the ℓ_∞-norm (the values of $R_{\max}$). In what follows, we overload notation by defining $R_{p,\ell}$ to denote $R_{g,\ell}$ where $g(r) = r^p$:

$$R_{p,\ell}(f) := \sum_{k=1}^{K} \left(\sum_{i=1}^{I} \ell\Big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\Big)\right)^p.$$

Thus, $R_{p,\ell}^{1/p}(f) \to R_{\max,\ell}(f)$ as $p \to \infty$, where $R_{\max,\ell}(f) := \max_k \sum_{i=1}^{I} \ell\big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\big)$.

As we will discuss, the choice of p should depend on the number of examples. More examples are needed for generalization if a larger value of p is chosen.

3. Illustrating That It Works

In this section, we will give some examples to illustrate how the objective concentrates on the top of the list when p is large, or more generally, when g is steep.

3.1 First Illustration: Swap on the Bottom vs. Swap on the Top

For our first illustration, we aim simply to show that the objective function we have derived really does care more about the top of the list than the rest. Consider the set of examples $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_8$ with vector of labels:

$$(-1, +1, -1, +1, -1, -1, +1, +1).$$

Consider scoring function f_{orig} which gives the scores: $f_{\text{orig}}(\mathbf{x}_i) = i$ for all i. Placing the labels in rank order of f_{orig} yields:

$$\text{labels in original rank order: } (-1 \quad +1 \quad -1 \quad +1 \quad -1 \quad -1 \quad +1 \quad +1) \ .$$

Using the power law $g(r) = r^4$ for the price function, we can compute the value of $R_{4,\mathbf{1}}(f_{\text{orig}})$ for this ranked list: $0^4 + 1^4 + 2^4 + 2^4 = 33$.

Now consider $f_{\text{swapOnBot}}$ which swaps the scores of a pair of examples at the bottom of the ranked list, $f_{\text{swapOnBot}}(\mathbf{x}_1) = 2, f_{\text{swapOnBot}}(\mathbf{x}_2) = 1$, and $f_{\text{swapOnBot}}(\mathbf{x}_i) = i$ for all other i. The new rank ordering of labels is:

$$\text{swap on the bottom: } (+1 \quad -1 \quad -1 \quad +1 \quad -1 \quad -1 \quad +1 \quad +1) \ .$$

Here a negative example is ranked above one more positive example than before. Computing the value of $R_{4,\mathbf{1}}(f_{\text{swapOnBot}})$ yields $1^4 + 1^4 + 2^4 + 2^4 = 34 > 33$; the value of $R_{4,\mathbf{1}}$ changes slightly when a swap is made at the bottom of the list, only from 33 to 34. Let us now instead consider a swap near the top of the list, so that the new set of labels is again only one swap away from the original, $f_{\text{swapOnTop}}(\mathbf{x}_6) = 7, f_{\text{swapOnTop}}(\mathbf{x}_7) = 6$, and $f_{\text{swapOnTop}}(\mathbf{x}_i) = i$ for all other i. The new ordering of labels is:

$$\text{swap on the top: } (-1 \quad +1 \quad -1 \quad +1 \quad -1 \quad +1 \quad -1 \quad +1) \ .$$

Here, the value of $R_{4,\mathbf{1}}(f_{\text{swapOnTop}})$ is $0^4 + 1^4 + 2^4 + 3^4 = 98 \gg 33$. So, in both cases only one swap was made between neighboring examples; however, the swap at the top of the list changed the objective dramatically (from 33 to 98) while the swap at the bottom hardly changed the objective at all (from 33 to 34). So, we have now illustrated that the objective function $R_{p,\mathbf{1}}(f)$ concentrates at the top of the list.

The same behavior occurs using different loss functions ℓ. This is summarized in Table 1 for three loss functions: the 0-1 loss which we have just explained, the exponential loss $\ell(r) = e^{-r}$, and the logistic loss $\ell(r) = \log(1 + e^{-r})$. (Note that using natural log for the logistic loss does not give an upper bound on the 0-1 loss, it is off by a multiplicative factor that is irrelevant in experiments.)

$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8$ y: $(-1,+1,-1,+1,-1,-1,+1,+1)$	$R_{4,\mathbf{1}}(f)$	$R_{4,\exp}(f)$	$R_{4,\text{logistic}}(f)$
labels ordered by f_{orig} : $(-1,+1,-1,+1,-1,-1,+1,+1)$	33	17,160.17	430.79
labels ordered by $f_{\text{swapOnBot}}$: $(+1,-1,-1,+1,-1,-1,+1,+1)$	34	72,289.39	670.20
labels ordered by $f_{\text{swapOnTop}}$: $(-1,+1,-1,+1,-1,+1,-1,+1)$	98	130,515.09	1,212.23

Table 1: Values of the objective function $R_{4,\ell}$ for the three slightly different labelings, using the 0-1 loss (column $R_{4,\mathbf{1}}$), exponential loss (column $R_{4,\exp}$), and logistic loss (column $R_{4,\text{logistic}}$). The objective functions change much more in reaction to the swap at the top of the list: the values in the third row (swap on the top) are significantly higher than those in the second row (swap on the bottom).

3.2 A Second Illustration: Reversal of Polarity

Let us assume we want to choose a scoring function f by minimizing our objective $R_{p,\ell}(f)$ over $f \in \mathcal{F}$ where $\mathcal{F}$ has only two functions, $\mathcal{F} = \{f_1, f_2\}$. This is an interesting experiment in which there are only 2 choices available for the function f: the first concentrates on the top of the ranked list, but performs poorly on the rest, whereas the second performs badly on the top of the ranked list, but performs well over all. In fact, the second scoring function f_2 is exactly a negation of the first scoring function f_1. Here are the labels and hypotheses:

labels	$+1$	$+1$	-1	-1	-1	-1	-1	$+1$	$+1$	$+1$	$+1$	$+1$	-1	-1	
f_1 :	(14	13	12	11	10	9	8	7	6	5	4	3	2	1	)/14
f_2 :	$(-14$	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	)/14

Here, f_1 performs well at the top of the list (the two top-scoring examples are positive), but the whole middle of the list is reversed; there are 5 negative examples in a row, and below that 5 positives. On the other hand, f_2 misses the top two examples which have scores -1/14 and -2/14, however, the 10 middle examples are correctly ranked. f_2 has a larger AUC than f_1, but f_1 is better at the top of the list. Now, which of f_1 and f_2 would the misranking objectives from Section 2 prefer? Let us answer this for various $R_{p,\ell}$, for different p and ℓ. Specifically, we will demonstrate that as p becomes larger, $R_{p,\ell}$ prefers the first hypothesis which performs better at the top. Table 2 shows values of $R_{p,\ell}$ for three different loss functions and for various values of p. This table shows that for smaller p, f_2 is preferred. At some value of p, the "polarity" reverses and then f_1 is preferred. So, using steeper price functions means that we are more likely to prefer scoring functions that perform well at the top of the list.

p	$R_{p,\mathbf{1}}(f_1)$	$R_{p,\mathbf{1}}(f_2)$	$\operatorname{argmin}_{f\in\{f_1,f_2\}} R_{p,\mathbf{1}}(f)$	$R_{p,\exp}(f_1)$	$R_{p,\exp}(f_2)$	$\operatorname{argmin}_{f\in\{f_1,f_2\}} R_{p,\exp}(f)$
1	25	24	f_2	50.25	49.80	f_2
2	125	118	f_2	367.39	362.35	f_2
3	625	726	f_1	$2.73*10^3$	$2.70*10^3$	f_2
4	$3.13*10^3$	$4.88*10^3$	f_1	$2.056*10^4$	$2.057*10^4$	f_1
5	$1.56*10^4$	$3.38*10^4$	f_1	$1.56*10^5$	$1.60*10^5$	f_1
6	$7.81*10^4$	$23.56*10^4$	f_1	$1.20*10^6$	$1.28*10^6$	f_1
7	$3.91*10^5$	$16.48*10^5$	f_1	$9.34*10^6$	$10.36*10^6$	f_1
8	$1.95*10^6$	$11.53*10^6$	f_1	$7.29*10^7$	$8.53*10^7$	f_1
9	$9.77*10^6$	$80.71*10^6$	f_1	$5.72*10^8$	$7.13*10^8$	f_1
10	$4.88*10^7$	$56.50*10^7$	f_1	$4.50*10^9$	$6.02*10^9$	f_1

p	$R_{p,\text{logistic}}(f_1)$	$R_{p,\text{logistic}}(f_2)$	$\operatorname{argmin}_{f\in\{f_1,f_2\}} R_{p,\text{logistic}}(f)$
1	34.34	34.09	f_2
2	170.18	167.90	f_2
3	851.09	836.46	f_2
4	$4.29*10^3$	$4.22*10^3$	f_2
5	$2.18*10^4$	$2.15*10^4$	f_2
6	$1.114*10^5$	$1.110*10^5$	f_2
7	$5.72*10^5$	$5.79*10^5$	f_1
8	$2.96*10^6$	$3.05*10^6$	f_1
9	$1.53*10^7$	$1.63*10^7$	f_1
10	$7.98*10^7$	$8.74*10^7$	f_1

Table 2: This table shows that as the price function gets steeper (as p increases), the scoring function f_1 that performs better on the top of the list is preferred. We show the values for each of the objectives $R_{p,\mathbf{1}}, R_{p,\exp}$ and $R_{p,\text{logistic}}$ for $p = 1,\ldots,10$ applied to f_1 (first column) and f_2 (second column). The third column shows which of the two scoring functions f_1 or f_2 achieve a lower value of the objective.

3.3 Third Illustration: Contribution of Each Positive-Negative Pair

Consider the following list of labels and function values :

$$\begin{array}{rl} y: & (1\ \ 1\ -1\ \ 1\ \ 1\ -1\ \ 1\ \ 1\ -1\ \ 1\ \ 1\ -1\ \ 1\ \ 1\ -1\ -1\ -1) \\ f: & (20\ 19\ 18\ 17\ 16\ 15\ 14\ 13\ 12\ 11\ 10\ \ 9\ \ 8\ \ 7\ \ 6\ \ 5\ \ 4\ \ 3\ \ 2\ \ 1)/20 \end{array}$$

Figure 1 illustrates the amount that each positive-negative pair contributes to $R_{p,\exp}$ for various values of p. We aim to show that $R_{p,\exp}$ becomes more influenced by the highest scoring negative examples as p is increased. On the vertical axis are the positive examples $i = 1,\ldots,12$ ordered by score, with the highest scoring examples at the bottom. On the horizontal axis are the negative examples $k = 1,\ldots,8$ ordered by score, with the highest scoring examples on the

left. The value of the $(i,k)^{th}$ entry is the contribution of the k^{th} highest scoring negative example, $\left(\sum_{\bar{i}} e^{-(f(\mathbf{x}_{\bar{i}})-f(\tilde{\mathbf{x}}_k))}\right)^p$, multiplied by the proportion attributed to the i^{th} highest scoring positive example, $e^{-(f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k))} / \sum_{\bar{i}} e^{-(f(\mathbf{x}_{\bar{i}})-f(\tilde{\mathbf{x}}_k))}$. As we adjust the value of p, one can see that most of the contribution shifts towards the left, or equivalently, towards the highest scoring negative examples.

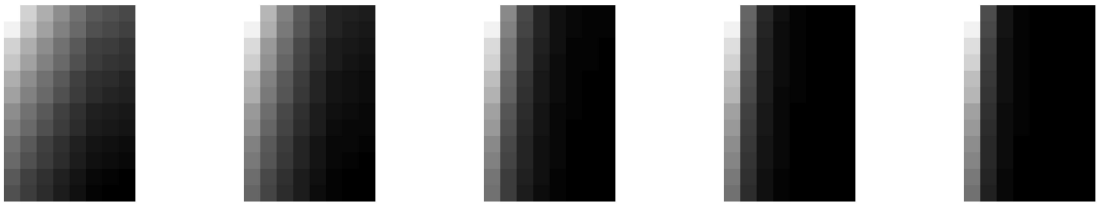


Figure 1: Contribution of each positive-negative pair to the objective $R_{p,\exp}$. Each square represents an i,k pair, where i is an index along the vertical axis, and k is along the horizontal axis. Lighter colors indicate larger contributions to $R_{p,\exp}$. The upper left corner represents the highest (worst) ranked negative and the lowest (worst) ranked positive.

4. A Generalization Bound for $R_{p,\mathbf{1}}$

We present two bounds, where the second has better dependence on p than the first. A preliminary version of the first bound appears in the conference version of this paper (Rudin, 2006). This work is inspired by the works of Koltchinskii and Panchenko (2002), Cucker and Smale (2002), and Bousquet (2003).

Assume that the positive instances $\mathbf{x}_i \in \mathcal{X}$, $i = 1,...,I$ are chosen independently and at random (iid) from a fixed but unknown probability distribution $\mathcal{D}_+$ on $\mathcal{X}$. Assume the negative instances $\tilde{\mathbf{x}}_k \in \mathcal{X}$, $k = 1,...,K$ are chosen iid from $\mathcal{D}_-$. The notation $\mathbf{x} \sim \mathcal{D}$ means $\mathbf{x}$ is chosen randomly according to distribution $\mathcal{D}$. The notation $S_+ \sim \mathcal{D}_+^I$ means each of the I elements of the training set S_+ are chosen independently at random according to $\mathcal{D}_+$. Similarly for $S_- \sim \mathcal{D}_-^K$.

We now define the "true" objective function for the underlying distribution:

$$\begin{aligned} R^{\text{true}}_{p,\mathbf{1}}(f) &:= \left(\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}\left(\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\mathbf{1}_{[f(\mathbf{x}_+)-f(\mathbf{x}_-)\le 0]}\right)^p\right)^{1/p} \\ &= \left\|\mathbb{P}_{\mathbf{x}_+\sim\mathcal{D}_+}\left(f(\mathbf{x}_+)-f(\mathbf{x}_-)\le 0|\mathbf{x}_-\right)\right\|_{L_p(\mathcal{X}_-,\mathcal{D}_-)}. \end{aligned}$$

The empirical loss associated with $R^{\text{true}}_{p,\mathbf{1}}(f)$ is the following:

$$R^{\text{empirical}}_{p,\mathbf{1}}(f) := \left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\mathbf{1}_{[f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k)\le 0]}\right)^p\right)^{1/p}.$$

Here, for a particular $\tilde{\mathbf{x}}_k$, $R^{\text{empirical}}_{p,\mathbf{1}}(f)$ takes into account the average number of positive examples that have scores below $\tilde{\mathbf{x}}_k$. It is a monotonic function of $R_{p,\mathbf{1}}$. To make this notion more general, let

us consider the average number of positive examples that have scores that are *close to* or below $\tilde{\mathbf{x}}_k$. A more general version of $R^{\text{empirical}}_{p,\mathbf{1}}(f)$ is thus defined as:

$$R^{\text{empirical}}_{p,\mathbf{1},\theta}(f) := \left(\frac{1}{K} \sum_{k=1}^{K} \left(\frac{1}{I} \sum_{i=1}^{I} \mathbf{1}_{[f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k) \leq \theta]} \right)^p \right)^{1/p}.$$

This terminology incorporates the "margin" value θ. As before, we suffer some loss whenever positive example $\mathbf{x}_i$ is ranked below negative example $\tilde{\mathbf{x}}_k$, but now we also suffer loss whenever $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_k$ have scores within θ of each other. Note that $R^{\text{empirical}}_{p,\mathbf{1},\theta}$ is an empirical quantity, so it can be measured for any θ. We will state two bounds, proved in Section 10, where the second is tighter than the first. The first bound is easier to understand and is a direct corollary of the second bound.

Theorem 2 *(First Generalization Bound) For all $\varepsilon > 0, p \geq 1, \theta > 0$, the probability over random choice of training set, $S_+ \sim \mathcal{D}_+^I, S_- \sim \mathcal{D}_-^K$ that there exists an $f \in \mathcal{F}$ such that*

$$R^{\text{true}}_{p,\mathbf{1}}(f) \geq R^{\textit{empirical}}_{p,\mathbf{1},\theta}(f) + \varepsilon$$

is at most:

$$2\mathcal{N}\left(\mathcal{F}, \frac{\varepsilon\theta}{8}\right) \left(\exp\left[-2 \left(\frac{\varepsilon}{4}\right)^{2p} K \right] + \exp\left[-\frac{\varepsilon^2}{8} I + \ln K \right] \right).$$

Here the covering number $\mathcal{N}(\mathcal{F}, \varepsilon)$ is defined as the number of ε-sized balls needed to cover $\mathcal{F}$ in L_∞, and it is used here as a complexity measure for $\mathcal{F}$. This expression states that, provided I and K are large, then with high probability, the true error $R^{\text{true}}_{p,\mathbf{1}}(f)$ is not too much more than the empirical error $R^{\text{empirical}}_{p,\mathbf{1},\theta}(f)$.

It is important to note the implications of this bound for scalability. More examples are required for larger p. This is because we are concentrating on a small portion of input space corresponding to the top of the ranked list. If most of the value of $R^{\text{true}}_{p,\mathbf{1}}$ comes from a small portion of input space, it is necessary to have more examples in that part of the space in order to estimate its value with high confidence. The fact that more examples are required for large p can affect performance in practice. A 1-dimensional demonstration of this fact is given at the end of Section 10.

Theorem 2 shows that the dependence on p is important for generalization. The following theorem shows that in most circumstances, we have much better dependence on p. Specifically, the dependence can be shifted from $-\varepsilon^{2p}$ in the exponential to a factor related to $-\varepsilon^2 \left(\inf_f R^{\text{true}}_{p,\mathbf{1}}(f) \right)^{2(p-1)}$. The bound becomes much tighter than Theorem 2 when all hypotheses have a large enough true risk, that is, when $\inf_f R^{\text{true}}_{p,\mathbf{1}}(f)$ is large compared to ε.

Theorem 3 *(Second Generalization Bound) For all $\varepsilon > 0, p \geq 1, \theta > 0$, the probability over random choice of training set, $S_+ \sim \mathcal{D}_+^I, S_- \sim \mathcal{D}_-^K$ that there exists an $f \in \mathcal{F}$ such that*

$$R^{\text{true}}_{p,\mathbf{1}}(f) \geq R^{\textit{empirical}}_{p,\mathbf{1},\theta}(f) + \varepsilon$$

is at most:

$$2\mathcal{N}\left(\mathcal{F}, \frac{\varepsilon\theta}{8}\right) \left(\exp\left[-2K \max\left\{ \frac{\varepsilon^2}{16} \left(R_{p,\min}\right)^{2(p-1)}, \left(\frac{\varepsilon}{4}\right)^{2p} \right\} \right] + \exp\left[-\frac{\varepsilon^2}{8} I + \ln K \right] \right).$$

where $R_{p,\min} := \inf_{f \in \mathcal{F}} R^{\text{true}}_{p,\mathbf{1}}(f)$.

The proof is in Section 10. The dependence on p is now much better than in Theorem 2. It is possible that the bound can be tightened in other ways, for instance, to use a different type of covering number. For instance, one might use the "sloppy covering number" in Rudin and Schapire (2009)'s ranking bound, which is adapted from the classification bound of Schapire et al. (1998).

The purpose of Theorems 2 and 3 is to provide the theoretical justification required for our choice of objective, provided a sufficient number of training examples. Having completed this, let us now write an algorithm for minimizing that objective.

5. A Boosting-Style Algorithm

We now choose a specific form for our objective $R_{g,\ell}$ by choosing ℓ. We have already chosen g to be a power law, $g(r) = r^p$. From now on, ℓ will be the exponential loss $\ell(r) = e^{-r}$. One could just as easily choose another loss; we choose the exponential loss in order to compare with RankBoost. The objective when $p = 1$ is exactly that of RankBoost, whose global objective is $R_{1,\exp}$. Here is the objective function, $R_{p,\exp}$ for $p \geq 1$:

$$R_{p,\exp}(f) := \sum_{k=1}^{K} \left(\sum_{i=1}^{I} e^{-(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k))} \right)^p .$$

The function f is constructed as a linear combination of "weak rankers" or "ranking features," $\{h_j\}_{j=1,\ldots,n}$, with $h_j : \mathcal{X} \to [0,1]$ so that $f = \sum_j \lambda_j h_j$, where $\boldsymbol{\lambda} \in \mathcal{R}^n$. Thus, the hypothesis space $\mathcal{F}$ is the class of convex combinations of weak rankers. Our objective is now $R_{p,\exp}(\boldsymbol{\lambda})$:

$$R_{p,\exp}(\boldsymbol{\lambda}) := \sum_{k=1}^{K} \left(\sum_{i=1}^{I} e^{-\left(\sum_j \lambda_j h_j(\mathbf{x}_i) - \sum_j \lambda_j h_j(\tilde{\mathbf{x}}_k)\right)} \right)^p = \sum_{k=1}^{K} \left(\sum_{i=1}^{I} e^{-(\mathbf{M}\boldsymbol{\lambda})_{ik}} \right)^p ,$$

where we have rewritten in terms of a matrix $\mathbf{M}$, which describes how each individual weak ranker j ranks each positive-negative pair $\mathbf{x}_i, \tilde{\mathbf{x}}_k$; this will make notation significantly easier. Define an index set that enumerates all positive-negative pairs $\mathcal{C}_p = \{ik : i \in 1,\ldots,I, k \in 1,\ldots,K\}$ where index ik corresponds to the i^{th} positive example and the k^{th} negative example. Formally,

$$M_{ik,j} := h_j(\mathbf{x}_i) - h_j(\tilde{\mathbf{x}}_k).$$

The size of $\mathbf{M}$ is $|\mathcal{C}_p| \times n$. The notation $(\cdot)_a$ means the a^{th} index of the vector, that is,

$$(\mathbf{M}\boldsymbol{\lambda})_{ik} := \sum_{j=1}^{n} M_{ik,j}\lambda_j = \sum_{j=1}^{n} \lambda_j h_j(\mathbf{x}_i) - \lambda_j h_j(\tilde{\mathbf{x}}_k).$$

The function $R_{p,\exp}(\boldsymbol{\lambda})$ is convex in $\boldsymbol{\lambda}$. This is because $e^{-(\mathbf{M}\boldsymbol{\lambda})_{ik}}$ is a convex function of $\boldsymbol{\lambda}$, any sum of convex functions is convex, and a composition of an increasing convex function with a convex function is convex. (Note that $R_{p,\exp}(\boldsymbol{\lambda})$ is convex but not necessarily strictly convex.)

We now derive a boosting-style coordinate descent algorithm for minimizing $R_{p,\exp}$ as a function of $\boldsymbol{\lambda}$. At each iteration of the algorithm, the coefficient vector $\boldsymbol{\lambda}$ is updated. At iteration t, we denote the coefficient vector by $\boldsymbol{\lambda}_t$. There is much background material available on the convergence of similar coordinate descent algorithms (for instance, see Zhang and Yu, 2005). We start with the objective at iteration t:

$$R_{p,\exp}(\boldsymbol{\lambda}_t) := \sum_{k=1}^{K} \left(\sum_{i=1}^{I} e^{(-\mathbf{M}\boldsymbol{\lambda}_t)_{ik}} \right)^p .$$

We then compute the variational derivative along each "direction" and choose weak ranker j_t to have largest absolute variational derivative. The notation $\mathbf{e}_j$ means a vector of 0's with a 1 in the j^{th} entry.

$$j_t \in \operatorname*{argmax}_j \left[-\frac{dR_{p,\exp}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_j)}{d\alpha}\Big|_{\alpha=0} \right], \text{ where}$$

$$\frac{dR_{p,\exp}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_j)}{d\alpha}\Big|_{\alpha=0} = p \sum_{k=1}^{K} \left[\left(\sum_{i=1}^{I} e^{(-\mathbf{M}\boldsymbol{\lambda}_t)_{ik}} \right)^{p-1} \left(\sum_{i=1}^{I} -M_{ik,j} e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{ik}} \right) \right].$$

Define the vector $\mathbf{q}_t$ on pairs i,k as $q_{t,ik} := e^{(-\mathbf{M}\boldsymbol{\lambda}_t)_{ik}}$, and the weight vector $\mathbf{d}_t$ as $d_{t,ik} := q_{t,ik} / \sum_{ik} q_{t,ik}$. Our choice of j_t becomes (ignoring constant factors that do not affect the argmax):

$$\begin{aligned} j_t \quad &\in \quad \operatorname*{argmax}_j \sum_{k=1}^{K} \left[\left(\sum_{i=1}^{I} d_{t,ik} \right)^{p-1} \sum_{i=1}^{I} d_{t,ik} M_{ik,j} \right] \\ &= \quad \operatorname*{argmax}_j \sum_{ik} \tilde{d}_{t,ik} M_{ik,j}, \text{ where } \tilde{d}_{t,ik} = d_{t,ik} \left(\sum_{i'} d_{t,i'k} \right)^{p-1}. \end{aligned}$$

To update the coefficient of weak ranker j_t, we now perform a linesearch for the minimum of $R_{p,\exp}$ along the j_t^{th} direction. The distance to travel in the j_t^{th} direction, denoted α_t, solves $0 = \frac{dR_{p,\exp}(\boldsymbol{\lambda}_t + \alpha \mathbf{e}_{j_t})}{d\alpha}\Big|_{\alpha_t}$. Ignoring division by constants, this equation becomes:

$$0 = \sum_{k=1}^{K} \left[\left(\sum_{i=1}^{I} d_{t,ik} e^{-\alpha_t M_{ik,j_t}} \right)^{p-1} \left(\sum_{i=1}^{I} M_{ik,j_t} d_{t,ik} e^{-\alpha_t M_{ik,j_t}} \right) \right]. \tag{1}$$

The value of α_t can be computed analytically in some cases, for instance, when the weak rankers are binary-valued and $p = 1$ (this is RankBoost). Otherwise, we simply use a linesearch to solve this equation for α_t. To complete the algorithm, we set $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t}$. To avoid having to compute $\mathbf{d}_{t+1}$ directly from $\boldsymbol{\lambda}_t$, we can perform the update by:

$$d_{t+1,ik} = \frac{d_{t,ik} e^{-\alpha_t M_{ik,j_t}}}{z_t} \quad \text{where} \quad z_t := \sum_{ik} d_{t,ik} e^{-\alpha_t M_{ik,j_t}}.$$

The full algorithm is shown in Figure 2. This implementation is not optimized for very large data sets since the size of $\mathbf{M}$ is $|C_p| \times n$. Note that the weak learning part of this algorithm in Step 3(a), when written in this form, is the same as for AdaBoost and RankBoost. Thus, any current implementation of a weak learning algorithm for AdaBoost or RankBoost can be directly used for the P-Norm Push.

6. Uniqueness of the Minimizer

We now show that a function $f = \sum_j \lambda_j h_j$ (or limit of functions) minimizing our objective is unique in some sense. Since $\mathbf{M}$ is not required to be invertible (and often is not), we cannot expect to find a unique vector $\boldsymbol{\lambda}$; one may achieve the identical values of $(\mathbf{M}\boldsymbol{\lambda})_{ik}$ with different choices of $\boldsymbol{\lambda}$. It is also true that elements of $\boldsymbol{\lambda}_t$ may approach $\pm\infty$, and furthermore, elements of $\mathbf{M}\boldsymbol{\lambda}_t$ often approach

1. **Input:** $\{\mathbf{x}_i\}_{i=1,...,I}$ positive examples, $\{\tilde{\mathbf{x}}_k\}_{k=1,...,K}$ negative examples, $\{h_j\}_{j=1,...,n}$ weak classifiers, $t_{\max}$ number of iterations, p power.

2. **Initialize:** $\lambda_{1,j} = 0$ for $j = 1,...,n$, $d_{1,ik} = 1/IK$ for $i = 1,...,I, k = 1,...,K$ $M_{ik,j} = h_j(\mathbf{x}_i) - h_j(\tilde{\mathbf{x}}_k)$ for all i,k,j

3. **Loop for** $t = 1,...,t_{\max}$

   (a) $j_t \in \text{argmax}_j \sum_{ik} \tilde{d}_{t,ik} M_{ik,j}$ where $\tilde{d}_{t,ik} = d_{t,ik} \left(\sum_{i'} d_{t,i'k}\right)^{p-1}$

   (b) Perform a linesearch for α_t. That is, find a value α_t that solves (1).

   (c) $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t}$, where $\mathbf{e}_{j_t}$ is 1 in position j_t and 0 elsewhere.

   (d) $z_t = \sum_{ik} d_{t,ik} e^{-\alpha_t M_{ik,j_t}}$

   (e) $d_{t+1,ik} = d_{t,ik} e^{-\alpha_t M_{ik,j_t}} / z_t$ for $i = 1,...,I, k = 1,...,K$

4. **Output:** $\boldsymbol{\lambda}_{t_{\max}}$

Figure 2: Pseudocode for the "P-Norm Push" algorithm.

$+\infty$, so it would seem difficult to prove (or even define) uniqueness. A trick that comes in handy for such situations is to use the closure of the space $Q' := \{\mathbf{q}' \in \mathcal{R}_+^{IK} | q'_{ik} = e^{-(\mathbf{M}\boldsymbol{\lambda})_{ik}} \text{ for some } \boldsymbol{\lambda} \in \mathcal{R}^n\}$. The closure of Q' includes the limits where $\mathbf{M}\boldsymbol{\lambda}_t$ becomes infinite, and considers the linear combination of hypotheses $\mathbf{M}\boldsymbol{\lambda}$ rather than $\boldsymbol{\lambda}$ itself, so it does not matter whether $\mathbf{M}$ is invertible. With the help of convex analysis, we will be able to show that our objective function yields a unique minimizer in the closure of Q'. Here is our uniqueness theorem:

Theorem 4 *Define* $Q' := \{\mathbf{q}' \in \mathcal{R}_+^{IK} | q'_{ik} = e^{-(\mathbf{M}\boldsymbol{\lambda})_{ik}} \text{ for some } \boldsymbol{\lambda} \in \mathcal{R}^n\}$ *and define* $\bar{Q}'$ *as the closure of* Q' *in* $\mathcal{R}^{IK}$. *Then for* $p \geq 1$, *there is a unique* $\mathbf{q}'^* \in \bar{Q}'$ *where:*

$$\mathbf{q}'^* = \text{argmin}_{\mathbf{q}' \in \bar{Q}'} \sum_k \left(\sum_i q'_{ik} \right)^p .$$

Our uniqueness proof depends mainly on the theory of convex duality for a class of Bregman distances, as defined by Della Pietra et al. (2002). This proof is inspired by Collins et al. (2002) who have proved uniqueness of this type for AdaBoost. In the case of AdaBoost, the primal optimization problem corresponds to a minimization over relative entropy. Our case is more unusual and the primal is not a common function. The proof of Theorem 4 is located in Section 11.

7. Variations of the Objective and Relationship to Information Retrieval Measures

It is possible to use variations of our basic derivation in Section 2 to derive other specialized objectives. Some of these objectives are similar to current popular quality measurements from information retrieval (IR), such as the "discounted cumulative gain" (DCG) (Järvelin and Kekäläinen, 2000). A basic property of this quality measurement, and additionally the average precision (the mean of precision values), is that it is proportional to a sum over relevant documents (which are the

positive examples in this setting), and uses a discounting factor that decreases according to the rank of a relevant document. The discounting factor here is analogous to the price function. Let us use the framework we have developed to derive new quality measurements with these properties.

Our derivation in Section 2 is designed to push the highly ranked negative examples down. Rearranging this argument, we can also pull the positive examples up, using the "reverse height." The reverse height of positive example i is the number of negative examples ranked above it.

$$\text{Reverse Height}(i) := \sum_k \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}.$$

The reverse height is very similar to the rank used in the IR quality measurements. The reverse height only considers the relationship of the positives to the negatives, and disregards the relationship of positives to each other. Precisely, define:

$$\text{Rank}(i) := \sum_k \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]} + \sum_{\tilde{i}} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\mathbf{x}_{\tilde{i}})]} = \text{Reverse Height}(i) + \sum_{\tilde{i}} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\mathbf{x}_{\tilde{i}})]}.$$

The rank can often be substituted for the reverse height. For discounting factor $g : \mathcal{R}_+ \to \mathcal{R}_+$, consider the variations of our objective:

$$R_{g,\mathbf{1}}^{\text{Reverse Height}}(f) = \sum_i g\left(\text{Reverse Height}(i)\right) = \sum_i g\left(\sum_k \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}\right).$$

$$R_{g,\mathbf{1}}^{\text{Rank}}(f) = \sum_i g\left(\text{Rank}(i)\right) = \sum_i g\left(\sum_k \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]} + \sum_{\tilde{i}} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\mathbf{x}_{\tilde{i}})]}\right).$$

Then, one might maximize $R_{g,\mathbf{1}}^{\text{Rank}}$ for various g. The function g should achieve the largest values for the positive examples i that possess the smallest reverse heights or ranks, since those are at the top of the list. It should thus be a decreasing function with steep negative slope near the y-axis. Choosing $g(z) = 1/z$ gives the average value of $1/\text{rank}$. Choosing $g(z) = 1/\ln(1+z)$ gives the discounted cumulative gain:

$$\text{AveR}(f) = \sum_i \frac{1}{\text{Rank}(i)} = \sum_i \frac{1}{\sum_k \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]} + \sum_{\tilde{i}} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\mathbf{x}_{\tilde{i}})]}},$$

$$\text{DCG}(f) = \sum_i \frac{1}{\ln\left(1 + \text{Rank}(i)\right)} = \sum_i \frac{1}{\ln\left(1 + \sum_k \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]} + \sum_{\tilde{i}} \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\mathbf{x}_{\tilde{i}})]}\right)}.$$

Let us consider the practical implications of minimizing the negation of the DCG. The discounting function $1/\ln(1+z)$ is decreasing, but its negation is not convex so there is no optimization guarantee. This is true even if we incorporate the exponential loss since $-1/\ln(1+e^z)$ is not convex. The same observation holds for the AveR.

It is possible, however, to choose a different discounting factor that allows us to create a convex objective to minimize. Let us choose a discounting factor of $-\ln(1+z)$, which is similar to the discounting factors for the AveR and DCG in that it is decreasing and convex. Figure 3 illustrates these discounting factors. Using this new discounting factor, and using the reverse height rather than the rank (which is an arbitrary choice), we arrive at the following objective:

$$R_{g_{\text{IR}},\mathbf{1}}(f) := \sum_i \ln\left(1 + \sum_k \mathbf{1}_{[f(\mathbf{x}_i) \leq f(\tilde{\mathbf{x}}_k)]}\right),$$

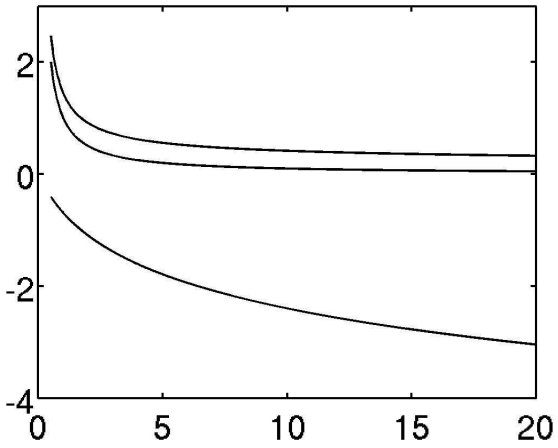


Figure 3: Discounting factor for discounted cumulative gain $1/\ln(1+z)$ (upper curve), discounting factor for the average of the reciprocal of the ranks $1/z$ (middle curve), and new discounting factor $-\ln(1+z)$ (lower curve) versus z.

and bounding the 0-1 loss from above,

$$R_{g_{\mathrm{IR}},\exp}(f) := \sum_i \ln\left(1+\sum_k e^{-(f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k))}\right). \qquad \text{“IR Push”} \tag{2}$$

Equation (2) is our version of IR-ranking measures, which we refer to by “IR Push” in Section 8. It is also very similar in essence to the objective for the multilabel problem defined by Dekel et al. (2004). The objective (2) is globally convex. In general, one must be careful when defining discounting factors in order to avoid non-convexity. Figure 4 illustrates the contribution of each positive-negative pair to $R_{g_{\mathrm{IR}},\exp}(f)$ for the set of labels and examples defined in Section 3.3. The slant towards the lower left indicates that this objective is biased towards the top of the list.

Concentrating on the Bottom: Since our objective concentrates at the top of the ranked list, it can just as easily be made to concentrate on the bottom of the ranked list by reversing the positive and negative examples, or equivalently, by using the reverse height with a discounting factor of $-z^p$. In this case, our p-norm objective becomes:

$$R^{\mathrm{Bottom}}_{p,\exp}(f) := \sum_{i=1}^{I}\left(\sum_{k=1}^{K} e^{-(f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k))}\right)^p.$$

Here, positive examples that score very badly are heavily penalized. $R^{\mathrm{Bottom}}_{p,\exp}(f)$ is also convex, so it can be easily minimized. Also, one can now write an objective that concentrates on the top and bottom simultaneously such as $R_{p,\exp}(f) + \text{const}\, R^{\mathrm{Bottom}}_{p,\exp}(f)$.

Crucial Pairs Formulation: The bipartite ranking problem is a specific case of the pairwise ranking problem. For the more general problem, the labels are replaced by a “truth function” $\pi : \mathcal{X} \times \mathcal{X} \rightarrow$

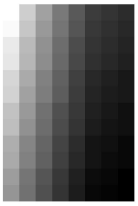


Figure 4: Contribution of each positive-negative pair to the objective $R_{g_{IR},\exp}$. Each square represents an i,k pair, where i is an index along the vertical axis, and k is along the horizontal axis, as described in Section 3.3. Lighter colors indicate larger contribution. The value of the i,k^{th} entry is the contribution of the i^{th} positive example, $\ln\left(1+\sum_k e^{-(f(\mathbf{x}_i)-f(\bar{\mathbf{x}}_k))}\right)$, multiplied by the proportion of the loss attributed to the k^{th} negative example, $e^{-(f(\mathbf{x}_i)-f(\bar{\mathbf{x}}_k))}/\sum_{\bar{k}} e^{-(f(\mathbf{x}_i)-f(\bar{\mathbf{x}}_{\bar{k}}))}$.

$\{0,1\}$, indicating whether the first element of the pair should be ranked above the second. In this case, one can replace the objective by:

$$R_{g,\ell}^{\text{Crucial Pairs}}(f) := \sum_{k=1}^{m} g\left(\sum_{i=1}^{m} \ell\Big(f(\mathbf{x}_i)-f(\mathbf{x}_k)\Big)\pi(\mathbf{x}_i,\mathbf{x}_k)\right),$$

where the indices i and k now run over all training examples. A slightly more general version of the above formula for $g(z)=z^p$ and the exponential loss was used by Ji et al. (2006) for the natural language processing problem of named entity recognition in Chinese. This algorithm performed quite well, in fact, within the margin of error of the best algorithm, but with a much faster training time. Its performance was substantially better than the support vector machine algorithm tested for this experiment. In Ji et al. (2006)'s setup, the P-Norm Push was used twice; the first time, a low value of p was chosen and a cutoff was made. The algorithm was used again for re-ranking (after some additional processing) with a higher value of p.

8. Experiments

The experiments of Ji et al. (2006) indicate the usefulness of our approach for larger, real-world problems. In this section, we will discuss the performance of the P-Norm Push on some smaller problems, since smaller problems are challenging when it comes to generalization. The choices we have made in Section 5 allow us to compare with RankBoost, which also uses the exponential loss. Furthermore, the choice of g as an adjustable power law allows us to illustrate the effect of the price g on the quality of the solution. Experiments have been performed using the P-Norm Push for $p=1$ (RankBoost), $2,4,8,16$ and 64, and using the IR Push information retrieval objective (2). For the P-Norm Push, the linesearch for α_t was performed using matlab's "fminunc" subroutine. The total number of iterations, $t_{\max}$, was fixed at 100 for all experiments. For the information retrieval objective, "fminunc" was used for the full optimization, which can be done for small experiments. Data were obtained from the UCI machine learning repository (Asuncion and Newman, 2007) and

all features were normalized to $[0,1]$. The three data sets chosen were MAGIC, ionosphere, and housing.

The first experiment uses the UCI MAGIC data set, which contains data from the Major Atmospheric Gamma Imaging Cherenkov Telescope project. The goal is to discriminate the statistical signatures of Monte Carlo simulated "gamma" particles from simulated "hadron" particles. In this problem, there are several relevant points on the ROC curve that determine the quality of the result. These points correspond to different acceptable false positive rates for different experiments, and all are close to the top of the list. There are 19020 examples (12332 gamma and 6688 hadron) and 11 features. Positive examples represent gamma particles and negative examples represent hadron particles. As a sample run, we chose 1000 examples randomly for training and tested on the rest.

Table 3 shows how different algorithms (the columns) performed with respect to different quality measures (the rows) on the MAGIC data. Each column of Table 3 represents a P-Norm Push or IR Push trial. The quality of the results is measured using the AUC (top row, larger values are better), $R_{p,\mathbf{1}}$ for various p (middle rows, smaller values are better), and the DCG and AveR (bottom rows, larger values are better). The best algorithms for each measure are summarized in bold and in the rightmost column. ROC curves and zoomed-in versions of the ROC curves for this sample run are shown in Figure 5. We expect the P-Norm Push for small p to yield the best results for

MAGIC data set

measure	p=1	p=2	p=4	p=8	p=16	p=64	IR	best
AUC	0.8370	**0.8402**	**0.8397**	0.8363	0.8329	0.8288	0.8284	small p
$R_{2,\mathbf{1}}$	6.5515	5.9731	5.5896	**5.4806**	**5.4990**	5.5819	5.5886	medium p
$R_{4,\mathbf{1}}$	4.2134	3.4875	2.8875	2.5638	2.4291	**2.3651**	**2.3582**	IR / large p
$R_{8,\mathbf{1}}$	3.8830	2.9138	2.1091	1.6266	1.3923	**1.2396**	**1.2257**	IR / large p
$R_{16,\mathbf{1}}$	6.8153	4.7208	3.0545	1.9698	1.4494	**1.1096**	**1.0823**	IR / large p
DCG	1.4022	1.4048	1.4066	1.4084	**1.4087**	**1.4087**	**1.4087**	IR / large p
AveR	8.1039	8.5172	8.6860	9.6701	**9.7520**	**9.7679**	**9.7688**	IR / large p

Table 3: Test performance of minimizers of $R_{p,\text{exp}}$ and $R_{g_{IR},\text{exp}}$ on a sample run with the MAGIC data set. Only significant digits are kept (factors of 10 have been removed). The best scores in each row are in bold and the right column summarizes the result by listing which algorithms performed the best with respect to each quality measure.

optimizing AUC, and we expect the large p and IR columns to yield the best results for $R_{p,\mathbf{1}}$ when p is large, and for the DCG and AveR. In other words, the rightmost column ought to say "small p" towards the top, followed by "medium p," and then "IR / large p." This general trend is observed. In this particular trial run, the IR Push and P-Norm Push for $p = 64$ yielded almost identical results, and their ROC curves are almost on top of each other in Figure 5.

The next experiment uses a much smaller data set, namely the UCI ionosphere data set, which has 351 examples (225 positive and 126 negative). These are data collected from a phased array of antennas. The goal is to distinguish "good" radar returns from "bad" radar returns. The good returns represent signals that reflect back towards the antenna, indicating structure in the ionosphere. The features are based on characteristics of the received signal. Out of the 34 features, we choose 5 of them (the last 5 features), which helps to alleviate overfitting, though there is still significant variation in results due to the small size of the data set. We used 3-fold cross-validation, where all

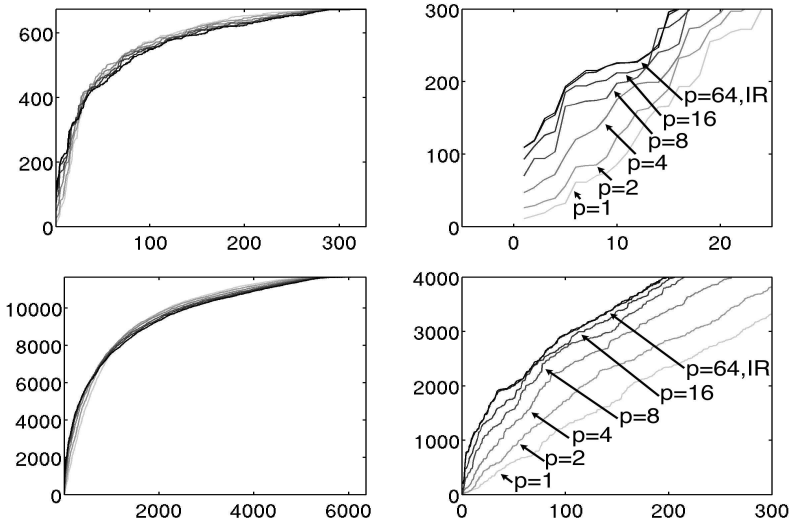


Figure 5: ROC Curves for the P-Norm Push and IR Push on the MAGIC data set. All plots are number of true positives vs. number of false positives. *Upper Left:* Full ROC Curves for training. *Upper Right:* Zoomed-in version of training ROC Curves. *Lower Left:* Full ROC Curves for testing. *Lower Right:* Zoomed-in version of testing ROC Curves.

algorithms were run once on each split, and the mean performance is reported in Table 4. ROC curves from one of the trials is presented in Figure 6. The trend from small to large p is able to be observed, despite variation due to train/test splits.

ionosphere data set

measure	p=1	p=2	p=4	p=8	p=16	p=64	IR	best
AUC	**0.6797**	0.6732	0.6700	0.6612	0.6479	0.6341	0.6409	small p
$R_{2,\mathbf{1}}$	2.1945	2.1931	2.1515	**2.1213**	2.1575	2.1974	2.1811	med/lg p
$R_{4,\mathbf{1}}$	2.0841	1.9891	1.8041	1.5911	1.4327	**1.3104**	1.4046	IR / large p
$R_{8,\mathbf{1}}$	3.7099	3.3459	2.6271	1.8950	1.3861	**1.0823**	1.2979	IR / large p
$R_{16,\mathbf{1}}$	1.7294	1.4558	0.8786	0.4236	0.2437	**0.1884**	0.2272	IR / large p
DCG	13.9197	14.1308	14.3261	14.5902	14.6916	**14.7903**	14.7169	IR / large p
AveR	2.9712	3.1610	3.3041	3.5084	3.5849	**3.6571**	3.6076	IR/ large p

Table 4: Mean test performance of minimizers of $R_{p,\text{exp}}$ and $R_{g_{IR},\text{exp}}$ over 3-fold cross-validation on the ionosphere data set.

We last consider the Boston Housing data set, which has 506 examples (35 positive, 471 negative), 13 features. This data set is skewed; there are significantly fewer positive examples than negative examples. In order to use the housing data set for a bipartite ranking problem, we used the fourth feature (which is binary) as the label y. The fourth feature describes whether a tract bounds the Charles River. Since there is some correlation between this feature and the other features

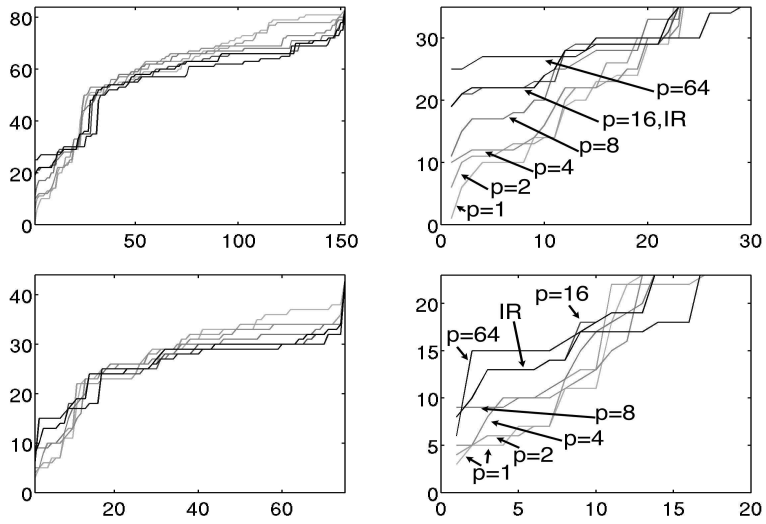


Figure 6: ROC Curves for the P-Norm Push and IR Push on ionosphere data set. All plots are number of true positives vs. number of false positives. *Upper Left:* Full ROC Curves for training. *Upper Right:* Zoomed-in version of training ROC Curves. *Lower Left:* Full ROC Curves for testing. *Lower Right:* Zoomed-in version of testing ROC Curves.

(such as distance to employment centers and tax-rate), it is reasonable for our learning algorithm to predict whether the tract bounds the Charles River based on the other features. We used 3-fold cross-validation (≈ 12 positives in each test set), where all algorithms were run once on each split, and the mean performance is reported in Table 5. ROC curves from one of the trials is presented in Figure 7. The trend from small to large p is again generally observed, despite variation due to data set size.

For all of these experiments, in agreement with our algorithm's derivation, a larger push (p large) causes the algorithm to perform better near the top of the ranked list on the training set. As discussed, this ability to correct the top of the list is not without sacrifice; we do sacrifice the ranks of items farther down on the list and we do reduce the value of the AUC, but we have made this choice on purpose in order to perform better near the top of the list.

9. Discussion and Open Problems

Here we describe interesting directions for future work.

9.1 Producing Dramatic Changes in the ROC curve

An open question is to quantify what properties of a hypothesis space and data set would allow an increase in p to cause a dramatic change in the ROC curve. In Section 8, we have shown cases where the benefits of increasing p are substantial, and in Section 3.2 we have shown that a dramatic

housing data set

measure	p=1	p=2	p=4	p=8	p=16	p=64	IR	best
AUC	**0.7739**	0.7633	0.7532	0.7500	0.7420	0.7330	0.7373	small p
$R_{2,\mathbf{1}}$	**3222**	3406	3665	3799	3818	3759	3847	small p
$R_{4,\mathbf{1}}$	**294078**	**292870**	304457	307135	305498	298611	304915	small/med p
$R_{8,\mathbf{1}}$	3.9056	3.5246	3.3908	3.2953	**3.2479**	3.3173	**3.2346**	IR / large p
$R_{16,\mathbf{1}}$	1.1762	0.9694	0.8337	0.8028	**0.7801**	0.8816	**0.7788**	IR / large p
DCG	3.6095	3.6476	3.6757	3.6858	**3.6977**	3.6671	**3.6931**	IR / large p
AveR	0.5241	0.5644	0.6022	0.6124	**0.6258**	0.6012	**0.6250**	IR / large p

Table 5: Mean test performance of minimizers of $R_{p,\exp}$ and $R_{g_{IR},\exp}$ over 3-fold cross validation with the housing data set. Only significant digits are kept (factors of 10 have been removed). The best scores in each row are in bold and the right column summarizes the result by listing which algorithms performed the best with respect to each quality measure.

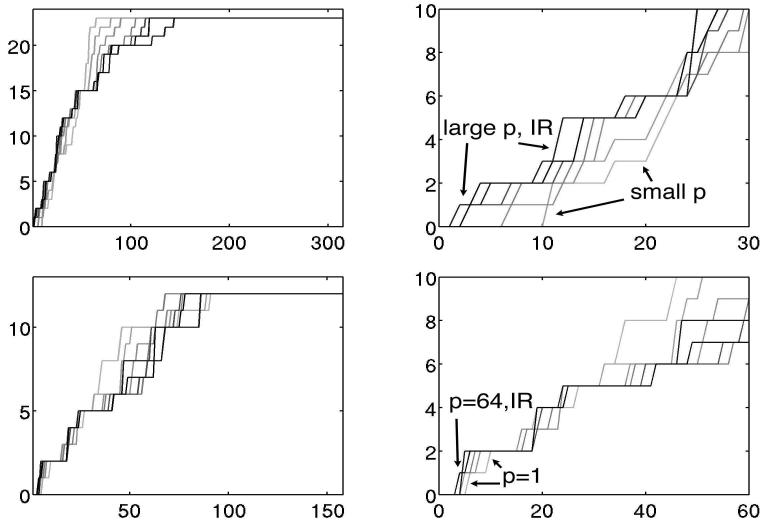


Figure 7: ROC Curves for the P-Norm Push and IR Push on the housing data set. All plots are number of true positives vs. number of false positives. *Upper Left:* Full ROC Curves for training. *Upper Right:* Zoomed-in version of training ROC Curves. *Lower Left:* Full ROC Curves for testing. *Lower Right:* Zoomed-in version of testing ROC Curves.

change is possible, even using an extremely small hypothesis space. However, it is sometimes the case that changes in p do not greatly affect the ROC curve.

A factor involved in this open question involves the flexibility of the hypothesis space with respect to the training set. Given a low capacity hypothesis space in which there is not too much flexibility in the set of solutions that yield good rankings, increasing p will not have much of an

effect. On the other hand, if a learning machine is high capacity, it probably has the flexibility to change the shape of the ROC curve dramatically. However, a high capacity learning machine generally is able to produce a consistent (or nearly consistent) ranking, and again, the choice of p probably does not have much effect. With respect to optimization on the training set, we have found the effect of increasing p to be the most dramatic when the hypothesis space is: limited (so as not to produce an almost consistent ranking), not too limited (features themselves are better than random guesses) and flexible (for instance, allowing some hypotheses to negate in order to produce a better solution as in Section 3.2). If such hypotheses are not available, we believe it is unlikely that any algorithm, whether the P-Norm Push, or any optimization algorithm for information retrieval measures, would be able to achieve a dramatic change in the ROC curve.

9.2 Optimizing $R_{\max}$ Directly

Given that there is no generalization guarantee for the ∞-norm, that is, $R_{\max}$, is it useful to directly minimize $R_{\max}$? This is still a convex optimization problem, and variations of this are done in other contexts, for instance, in the context of label ranking by Shalev-Shwartz and Singer (2006) and Crammer and Singer (2001). One might consider, for instance, optimizing $R_{\max}$ and measuring success on the test set using $R_{p,\mathbf{1}}$ for $p < \infty$.

One answer is provided by the equivalence of norms in finite dimensions. For instance, the value of $R_{\max}$ scales with $R_{p,\mathbf{1}}$, as demonstrated in Theorem 1. So optimizing $R_{\max}$ would still possibly be useful with respect to measuring success on smaller p (though in this case, one could optimize $R_{p,\ell}$).

9.3 Choices for ℓ and g

An important direction for future research is the choice of loss function ℓ and price function g. This framework is flexible in that different choices for ℓ and g can be chosen based on the particular goal, whether it is to optimize the AUC, $R_{p,\mathbf{1}}$ for some p, one of the IR measures suggested, or something totally different. The objective for the IR measures needed a concave price function $\ln(1+z)$, in which case the objective convex was made convex by using the exponential loss, in other words., $\ln(1+e^x)$ is convex. It may be possible to leverage the loss function in other cases, allowing us to consider more varied price functions while still working with an objective that is convex. One appealing possibility is to choose a non-monotonic function for g, which might allow us to concentrate on a specific portion of the ROC Curve; however, it may be difficult to maintain the convexity of the objective through the choice of the loss function.

Now we move on to the proofs.

10. Proof of Theorem 2 and Theorem 3

We define a Lipschitz function $\phi : \mathcal{R} \to \mathcal{R}$ (with Lipschitz constant $\mathrm{Lip}(\phi)$) which will act as our loss function, and gives us the margin. We will eventually use the same piecewise linear definition of ϕ as Koltchinskii and Panchenko (2002), but for now, we require only that ϕ obey $0 \le \phi(z) \le 1 \; \forall z$ and $\phi(z) = 1$ for $z < 0$. Since $\phi(z) \ge \mathbf{1}_{[z \le 0]}$, we can define an upper bound for $R^{\text{true}}_{p,\mathbf{1}}(f)$:

$$R^{\text{true}}_{p,\phi}(f) := \left(\mathbb{E}_{\mathbf{x}_- \sim \mathcal{D}_-} \left(\mathbb{E}_{\mathbf{x}_+ \sim \mathcal{D}_+} \phi\big(f(\mathbf{x}_+) - f(\mathbf{x}_-)\big) \right)^p \right)^{1/p}.$$

We have $R^{\text{true}}_{p,\mathbf{1}}(f) \leq R^{\text{true}}_{p,\phi}(f)$. The empirical error associated with $R^{\text{true}}_{p,\phi}$ is:

$$R^{\text{empirical}}_{p,\phi}(f) := \left(\frac{1}{K} \sum_{k=1}^{K} \left(\frac{1}{I} \sum_{i=1}^{I} \phi\big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\big) \right)^p \right)^{1/p}.$$

First, we bound from above the quantity $R^{\text{true}}_{p,\phi}$ by two terms: the empirical error term $R^{\text{empirical}}_{p,\phi}$, and a term characterizing the deviation of $R^{\text{empirical}}_{p,\phi}$ from $R^{\text{true}}_{p,\phi}$ uniformly:

$$\begin{aligned} R^{\text{true}}_{p,\mathbf{1}}(f) &\leq R^{\text{true}}_{p,\phi}(f) = R^{\text{true}}_{p,\phi}(f) - R^{\text{empirical}}_{p,\phi}(f) + R^{\text{empirical}}_{p,\phi}(f) \\ &\leq \sup_{\bar{f} \in \mathcal{F}} \left(R^{\text{true}}_{p,\phi}(\bar{f}) - R^{\text{empirical}}_{p,\phi}(\bar{f}) \right) + R^{\text{empirical}}_{p,\phi}(f). \end{aligned}$$

The proof of Theorem 3 mainly involves an upper bound on the first term. The second term will be upper bounded by $R^{\text{empirical}}_{p,\mathbf{1},\theta}(f)$ by our choice of ϕ. Define $L(f)$ as follows:

$$L(f) := R^{\text{true}}_{p,\phi}(f) - R^{\text{empirical}}_{p,\phi}(f).$$

Let us outline the proof that follows. The goal is to bound $L(f)$ uniformly over $f \in \mathcal{F}$. To do this, we use a covering number argument similar to that of Cucker and Smale (2002). First, we will cover $\mathcal{F}$ by L_∞ disks. We show in Lemma 5 (below) that the value of $L(f)$ within each disk does not change very much provided that the disks are small. We then derive a probabilistic bound on $L(f)$ for any f in Lemma 9, and use this bound on representatives f_r from each disk. A union bound over disks yields the result. The most effort of this proof is devoted to the bound on $L(f)$ in Lemma 9 below, which uses McDiarmid's Inequality. Let us now proceed with the proof.

The following lemma is true for every training set S. It will be used later to show that the value of $L(f)$ does not change much within each L_∞ ball.

Lemma 5 *For any two functions $f_1, f_2 \in L_\infty(\mathcal{X})$,*

$$L(f_1) - L(f_2) \leq 4\text{Lip}(\phi)||f_1 - f_2||_\infty.$$

Proof First, we rearrange the terms:

$$\begin{aligned} L(f_1) - L(f_2) &= R^{\text{true}}_{p,\phi}(f_1) - R^{\text{empirical}}_{p,\phi}(f_1) - R^{\text{true}}_{p,\phi}(f_2) + R^{\text{empirical}}_{p,\phi}(f_2) \\ &= [R^{\text{true}}_{p,\phi}(f_1) - R^{\text{true}}_{p,\phi}(f_2)] - [R^{\text{empirical}}_{p,\phi}(f_1) - R^{\text{empirical}}_{p,\phi}(f_2)]. \end{aligned} \quad (3)$$

We bound from above the second bracketed term of (3),

$$\begin{aligned}
&R^{\text{empirical}}_{p,\phi}(f_1) - R^{\text{empirical}}_{p,\phi}(f_2) \\
&= \left[\frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{I}\sum_{i=1}^{I}\phi\Big(f_1(\mathbf{x}_i) - f_1(\tilde{\mathbf{x}}_k)\Big)\right]^p\right]^{1/p} - \left[\frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{I}\sum_{i=1}^{I}\phi\Big(f_2(\mathbf{x}_i) - f_2(\tilde{\mathbf{x}}_k)\Big)\right]^p\right]^{1/p} \\
&\leq \left[\frac{1}{K}\sum_{k=1}^{K}\left|\frac{1}{I}\sum_{i=1}^{I}\phi\Big(f_1(\mathbf{x}_i) - f_1(\tilde{\mathbf{x}}_k)\Big) - \frac{1}{I}\sum_{i=1}^{I}\phi\Big(f_2(\mathbf{x}_i) - f_2(\tilde{\mathbf{x}}_k)\Big)\right|^p\right]^{1/p} \\
&\leq \left[\frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{I}\sum_{i=1}^{I}\left|\phi\Big(f_1(\mathbf{x}_i) - f_1(\tilde{\mathbf{x}}_k)\Big) - \phi\Big(f_2(\mathbf{x}_i) - f_2(\tilde{\mathbf{x}}_k)\Big)\right|\right]^p\right]^{1/p} \\
&\leq \left[\frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{I}\sum_{i=1}^{I}\text{Lip}(\phi)\Big|f_1(\mathbf{x}_i) - f_1(\tilde{\mathbf{x}}_k) - f_2(\mathbf{x}_i) + f_2(\tilde{\mathbf{x}}_k)\Big|\right]^p\right]^{1/p} \\
&\leq \left[\frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{I}\sum_{i=1}^{I}\text{Lip}(\phi)2\sup_{\mathbf{x}}\Big|f_1(\mathbf{x}) - f_2(\mathbf{x})\Big|\right]^p\right]^{1/p} = 2\text{Lip}(\phi)\|f_1 - f_2\|_\infty.
\end{aligned}$$

Here, we have used Minkowski's inequality for $\ell_p(\mathcal{R}^K)$, which is the triangle inequality $\|f-g\|_p \geq \|f\|_p - \|g\|_p$, and the definition of the Lipschitz constant for ϕ. An identical calculation for the first bracketed term of (3), again using Minkowski's inequality yields:

$$R^{\text{true}}_{p,\phi}(f_1) - R^{\text{true}}_{p,\phi}(f_2) \leq 2\text{Lip}(\phi)||f_1 - f_2||_\infty.$$

Combining the two terms yields the statement of the lemma. ■

The following step appears in Cucker and Smale (2002). Let $\ell_\varepsilon := \mathcal{N}\left(\mathcal{F}, \frac{\varepsilon}{8\text{Lip}(\phi)}\right)$, the covering number of $\mathcal{F}$ by L_∞ disks of radius $\frac{\varepsilon}{8\text{Lip}(\phi)}$. Define $f_1, f_2, ..., f_{\ell_\varepsilon}$ to be the centers of such a cover. In other words, the collection of L_∞ disks B_r centered at f_r and with radius $\frac{\varepsilon}{8\text{Lip}(\phi)}$ is a cover for $\mathcal{F}$. In the proof of the theorem, we will use the center of each disk to act as a representative for the whole disk. So, we must show that we do not lose too much by using f_r as a representative for disk B_r.

Lemma 6 *For all* $\varepsilon > 0$,

$$\mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left\{\sup_{f\in B_r} L(f) \geq \varepsilon\right\} \leq \mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left\{L(f_r) \geq \frac{\varepsilon}{2}\right\}.$$

Proof By Lemma 5, for every training set S and for all $f \in B_r$,

$$\sup_{f\in B_r} L(f) - L(f_r) \leq 4\text{Lip}(\phi)\sup_{f\in B_r}||f - f_r||_\infty \leq 4\text{Lip}(\phi)\frac{\varepsilon}{8\text{Lip}(\phi)} = \frac{\varepsilon}{2}$$

Thus,

$$\sup_{f\in B_r} L(f) \geq \varepsilon \implies L(f_r) \geq \frac{\varepsilon}{2}.$$

The statement of the lemma follows directly. ■

Here is an inequality that will be useful in the next proof as the mechanism for incorporating p into the bound.

Lemma 7 *For $0 \leq a, b \leq 1$,*

$$|a^{1/p} - b^{1/p}| \leq \min\left\{ |a-b|a^{(1/p)-1}, |a-b|^{1/p} \right\}.$$

Proof For $p = 1$ there is nothing to prove, so take $p > 1$. We need to show both

$$|a^{1/p} - b^{1/p}| \leq |a-b|a^{(1/p)-1} \tag{4}$$

and

$$|a^{1/p} - b^{1/p}| \leq |a-b|^{1/p}. \tag{5}$$

Let us show (5) first. For $z_1, z_2 \geq 0$, it is true that $z_1^p + z_2^p \leq (z_1 + z_2)^p$ as an immediate consequence of the binomial theorem. When $a \geq b$, substitute $z_1 = (a-b)^{1/p}$, $z_2 = b^{1/p}$. The result follows after simplification. The case $a \leq b$ is completely symmetric so no additional work is needed. To show (4), consider first the case $a \geq b$, so that

$$b^{1/p-1} \geq a^{1/p-1}.$$

Multiplying by b yields $b^{1/p} \geq a^{1/p-1}b$, negating and adding $a^{1/p}$ yields

$$a^{1/p} - b^{1/p} \leq a^{1/p} - a^{1/p-1}b, \text{ so } a^{1/p} - b^{1/p} \leq (a-b)a^{1/p-1}.$$

Exactly the same steps (with reversed inequalities) can be used to show the $b \geq a$ case. ■

The benefit of using the minimum in Lemma 7 is that the first term most often gives a tighter bound. In the case where it does not do so, the second term applies. An illustration of this inequality is provided in Figure 8.

We now incorporate the fact that the training set is chosen randomly. We will use a generalization of Hoeffding's inequality due to McDiarmid, as follows:

Theorem 8 ***(McDiarmid, 1989)*** *Let $X_1, X_2, ...X_m$ be independent random variables under distribution D on $\mathcal{X}$. Let $f : \mathcal{X}^m \to \mathcal{R}$ be any function such that:*

$$\sup_{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m, \mathbf{x}_i'} \left| f(\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_m) - f(\mathbf{x}_1, \ldots, \mathbf{x}_i', \ldots \mathbf{x}_m) \right| \leq c_i \quad \textit{for} \quad 1 \leq i \leq m.$$

Then for any $\varepsilon > 0$,

$$\begin{aligned}
\mathbb{P}_{X_1, X_2, \ldots, X_m \sim D}\left\{ f(X_1, X_2, ..., X_m) - \mathbb{E}\left[f(X_1, X_2, ..., X_m) \right] \geq \varepsilon \right\} &\leq \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^m c_i^2} \right), \\
\mathbb{P}_{X_1, X_2, \ldots, X_m \sim D}\left\{ \mathbb{E}\left[f(X_1, X_2, ..., X_m) \right] - f(X_1, X_2, ..., X_m) \geq \varepsilon \right\} &\leq \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^m c_i^2} \right),
\end{aligned}$$

and thus by the union bound,

$$\mathbb{P}_{X_1, X_2, \ldots, X_m \sim D}\left\{ \left| f(X_1, X_2, ..., X_m) - \mathbb{E}\left[f(X_1, X_2, ..., X_m) \right] \right| \geq \varepsilon \right\} \leq 2\exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^m c_i^2} \right).$$

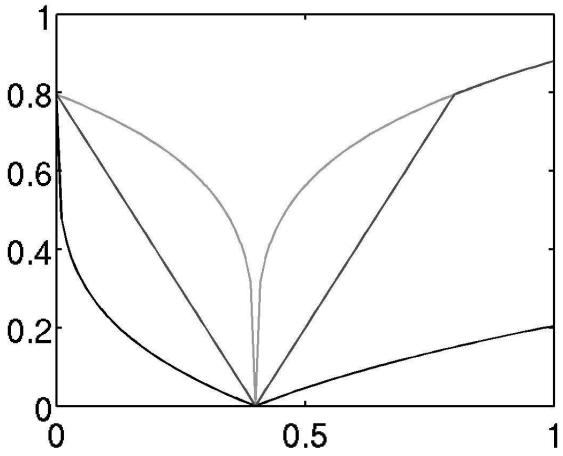


Figure 8: Functions $|a^{1/p} - b^{1/p}|$ (lower curve), $|a-b|^{1/p}$ (upper curve), and $\min(|a-b|a^{(1/p)-1}, |a-b|^{1/p})$ (middle curve) versus b. For this figure $p = 4$ and $a = 0.4$. One can see that in most cases, $|a-b|a^{(1/p)-1}$ is a better approximation to $|a^{1/p} - b^{1/p}|$ than $|a-b|^{1/p}$).

Here is our main probabilistic bound on $L(f)$ for an individual f. It uses McDiarmid's Inequality (Theorem 8) and Lemma 7.

Lemma 9 *For all $\varepsilon_1 > 0$, for all $f \in \mathcal{F}$:*

$$\begin{aligned}
&\mathbb{P}_{S_+ \sim \mathcal{D}_+^I, S_- \sim \mathcal{D}_-^K} \left(L(f) \geq \varepsilon_1 \right) \\
&\quad \leq \quad 2\exp\left[-2K \max\left\{ \frac{\varepsilon_1^2}{4} \left(R_{p,\phi}^{\text{true}}(f) \right)^{2(p-1)}, \left(\frac{\varepsilon_1}{2} \right)^{2p} \right\} \right] + 2\exp\left[-\frac{\varepsilon_1^2}{2} I + \ln K \right]. \qquad (6)
\end{aligned}$$

Proof Define

$$R_{p,\phi}^{\mathcal{DS}}(f) := \left(\frac{1}{K} \sum_{k=1}^{K} \left(\mathbb{E}_{\mathbf{x}_+ \sim \mathcal{D}_+} \phi\Big(f(\mathbf{x}_+) - f(\tilde{\mathbf{x}}_k) \Big) \right)^p \right)^{1/p}.$$

Now,

$$\begin{aligned}
&\mathbb{P}_{S_+ \sim \mathcal{D}_+^I, S_- \sim \mathcal{D}_-^K} \Big(L(f) \geq \varepsilon_1 \Big) \\
&\quad = \quad \mathbb{P}_{S_+ \sim \mathcal{D}_+^I, S_- \sim \mathcal{D}_-^K} \Big(R_{p,\phi}^{\text{true}}(f) - R_{p,\phi}^{\mathcal{DS}}(f) + R_{p,\phi}^{\mathcal{DS}}(f) - R_{p,\phi}^{\text{empirical}}(f) \geq \varepsilon_1 \Big) \\
&\quad \leq \quad \mathbb{P}_{S_- \sim \mathcal{D}_-^K} \left(R_{p,\phi}^{\text{true}}(f) - R_{p,\phi}^{\mathcal{DS}}(f) \geq \frac{\varepsilon_1}{2} \right) + \mathbb{P}_{S_+ \sim \mathcal{D}_+^I, S_- \sim \mathcal{D}_-^K} \left(R_{p,\phi}^{\mathcal{DS}}(f) - R_{p,\phi}^{\text{empirical}}(f) \geq \frac{\varepsilon_1}{2} \right) \\
&\quad =: \quad \text{term}_1 + \text{term}_2. \qquad (7)
\end{aligned}$$

We bound term_1 and term_2 of (7) separately.

Bound on term$_1$*:* The following uses Lemma 7 above (translating into notation of the lemma):

$$\begin{aligned}
&R^{\text{true}}_{p,\phi}(f) - R^{\mathcal{D}S}_{p,\phi}(f) \\
&= \left(\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}\left(\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_+) - f(\mathbf{x}_-)\Big)\right)^p\right)^{1/p} \\
&\quad - \left(\frac{1}{K}\sum_{k=1}^{K}\left(\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\Big)\right)^p\right)^{1/p} \\
&= a^{1/p} - b^{1/p} \leq |a^{1/p} - b^{1/p}| \leq \min\left\{|a-b|a^{(1/p)-1}, |a-b|^{1/p}\right\}.
\end{aligned}$$

Thus for all $\varepsilon_1 > 0$,

$$\begin{aligned}
&\mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left(R^{\text{true}}_{p,\phi}(f) - R^{\mathcal{D}S}_{p,\phi}(f) \geq \frac{\varepsilon_1}{2}\right) \\
&\leq \mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left(\min\left\{|a-b|a^{(1/p)-1}, |a-b|^{1/p}\right\} \geq \frac{\varepsilon_1}{2}\right) \\
&= \mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left(|a-b|a^{1/p-1} \geq \frac{\varepsilon_1}{2} \bigcap |a-b|^{\frac{1}{p}} \geq \frac{\varepsilon_1}{2}\right) \\
&= \mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left(|a-b| \geq \frac{\varepsilon_1}{2}a^{1-1/p} \bigcap |a-b| \geq \left(\frac{\varepsilon_1}{2}\right)^p\right) \\
&= \mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left(|a-b| \geq \frac{\varepsilon_1}{2}\left(R^{\text{true}}_{p,\phi}(f)\right)^{p-1} \bigcap |a-b| \geq \left(\frac{\varepsilon_1}{2}\right)^p\right) \\
&= \mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left(|a-b| \geq \max\left\{\frac{\varepsilon_1}{2}\left(R^{\text{true}}_{p,\phi}(f)\right)^{p-1}, \left(\frac{\varepsilon_1}{2}\right)^p\right\}\right).
\end{aligned}$$

Let

$$\varepsilon_2 := \max\left\{\frac{\varepsilon_1}{2}\left(R^{\text{true}}_{p,\phi}(f)\right)^{p-1}, \left(\frac{\varepsilon_1}{2}\right)^p\right\}.$$

Then,

$$\begin{aligned}
&\mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left(R^{\text{true}}_{p,\phi}(f) - R^{\mathcal{D}S}_{p,\phi}(f) \geq \frac{\varepsilon_1}{2}\right) \\
&\leq \mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left(\left|\mathbb{E}_{\mathbf{x}_-\sim\mathcal{D}_-}\left(\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_+) - f(\mathbf{x}_-)\Big)\right)^p - \right.\right. \\
&\qquad \left.\left.\frac{1}{K}\sum_{k=1}^{K}\left(\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_+) - f(\tilde{\mathbf{x}}_k)\Big)\right)^p\right| \geq \varepsilon_2\right).
\end{aligned}$$

The largest possible change in $\frac{1}{K}\sum_{k=1}^{K}\left(\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_+) - f(\tilde{\mathbf{x}}_k)\Big)\right)^p$ due to the replacement of one negative example is $1/K$. By McDiarmid's inequality,

$$\begin{aligned}
&\mathbb{P}_{S_-\sim\mathcal{D}_-^K}\left(R^{\text{true}}_{p,\phi}(f) - R^{\mathcal{D}S}_{p,\phi}(f) \geq \frac{\varepsilon_1}{2}\right) \\
&\leq \exp\left(-\frac{2\varepsilon_2^2}{K\frac{1}{K^2}}\right) = 2\exp(-2K\varepsilon_2^2) \\
&= 2\exp\left(-2K\max\left\{\frac{\varepsilon_1^2}{4}\left(R^{\text{true}}_{p,\phi}(f)\right)^{2(p-1)}, \left(\frac{\varepsilon_1}{2}\right)^{2p}\right\}\right). \qquad (8)
\end{aligned}$$

Bound on term$_2$:

$$\begin{aligned} & R^{\mathcal{D}S}_{p,\phi}(f) - R^{\text{empirical}}_{p,\phi}(f) \\ = & \left(\frac{1}{K}\sum_{k=1}^{K}\left(\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_+) - f(\mathbf{x}_-)\Big)\right)^p\right)^{1/p} \\ & - \left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\phi\Big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\Big)\right)^p\right)^{1/p}. \end{aligned}$$

Thus for all $\varepsilon_1 > 0$,

$$\begin{aligned} & \mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left(R^{\mathcal{D}S}_{p,\phi}(f) - R^{\text{empirical}}_{p,\phi}(f) \geq \frac{\varepsilon_1}{2}\right) \\ = & \mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left(\frac{1}{K^{1/p}}\left\|\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_+) - f(\cdot)\Big)\right\|_{\ell_p(R^K)}\right. \\ & \qquad \left. - \frac{1}{K^{1/p}}\left\|\frac{1}{I}\sum_{i=1}^{I}\phi\Big(f(\mathbf{x}_i) - f(\cdot)\Big)\right\|_{\ell_p(R^K)} \geq \frac{\varepsilon_1}{2}\right) \\ \leq & \mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left(\frac{1}{K^{1/p}}\left\|\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_+) - f(\cdot)\Big) - \frac{1}{I}\sum_{i=1}^{I}\phi\Big(f(\mathbf{x}_i) - f(\cdot)\Big)\right\|_{\ell_p(R^K)} \geq \frac{\varepsilon_1}{2}\right) \\ \leq & \mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left(\left\|\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_+) - f(\cdot)\Big) - \frac{1}{I}\sum_{i=1}^{I}\phi\Big(f(\mathbf{x}_i) - f(\cdot)\Big)\right\|_{\ell_\infty(R^K)} \geq \frac{\varepsilon_1}{2}\right) \\ = & \mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left(\exists k : \left|\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_+) - f(\tilde{\mathbf{x}}_k)\Big) - \frac{1}{I}\sum_{i=1}^{I}\phi\Big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\Big)\right| \geq \frac{\varepsilon_1}{2}\right). \end{aligned}$$

We now use McDiarmid's Inequality. The largest possible change in $\frac{1}{I}\sum_{i=1}^{I}\phi\Big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\Big)$ due to the replacement of one positive example is $1/I$. Thus, for all $\tilde{\mathbf{x}}_k$,

$$\begin{aligned} & \mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left(\left|\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_+) - f(\tilde{\mathbf{x}}_k)\Big) - \frac{1}{I}\sum_{i=1}^{I}\phi\Big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\Big)\right| \geq \frac{\varepsilon_1}{2}\right) \\ \leq & 2\exp\left[-\frac{2\left(\frac{\varepsilon_1}{2}\right)^2}{I\frac{1}{I^2}}\right] = 2\exp\left[-\frac{\varepsilon_1^2}{2}I\right]. \end{aligned}$$

By the union bound over the K negative examples:

$$\begin{aligned} & \mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left(\exists k : \left|\mathbb{E}_{\mathbf{x}_+\sim\mathcal{D}_+}\phi\Big(f(\mathbf{x}_+) - f(\tilde{\mathbf{x}}_k)\Big) - \frac{1}{I}\sum_{i=1}^{I}\phi\Big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\Big)\right| \geq \frac{\varepsilon_1}{2}\right) \\ \leq & 2K\exp\left[-\frac{\varepsilon_1^2}{2}I\right] = 2\exp\left[-\frac{\varepsilon_1^2}{2}I + \ln(K)\right], \end{aligned}$$

and thus,

$$\mathbb{P}_{S_+\sim\mathcal{D}_+^I, S_-\sim\mathcal{D}_-^K}\left(R^{\text{true}}_{p,\phi}(f) - R^{\mathcal{D}S}_{p,\phi}(f) \geq \frac{\varepsilon_1}{2}\right) \leq 2\exp\left[-\frac{\varepsilon_1^2}{2}I + \ln K\right].$$

Combining this with (8) and (7) yields the statement of the lemma. ■

Proof *(Of Theorem 2 and Theorem 3)* Since the B_r are a cover for $\mathcal{F}$, it is true that

$$\sup_{f\in\mathcal{F}} L(f) \geq \varepsilon \iff \exists r \leq \ell_\varepsilon \text{ such that } \sup_{f\in B_r} L(f) \geq \varepsilon.$$

First applying the union bound over balls, then applying Lemma 6 we find:

$$\begin{aligned}
&\mathbb{P}_{S_+\sim\mathcal{D}^I, S_-\sim\mathcal{D}^K}\left\{\sup_{f\in\mathcal{F}} L(f) \geq \varepsilon\right\} \\
&\leq \sum_{r=1}^{\ell_\varepsilon} \mathbb{P}_{S_+\sim\mathcal{D}^I, S_-\sim\mathcal{D}^K}\left\{\sup_{f\in B_r} L(f) \geq \varepsilon\right\} \\
&\leq \sum_{r=1}^{\ell_\varepsilon} \mathbb{P}_{S_+\sim\mathcal{D}^I, S_-\sim\mathcal{D}^K}\left\{L(f_r) \geq \varepsilon/2\right\}.
\end{aligned}$$

We bound from above using (6) in order to prove Theorem 3 using $\varepsilon_1 = \varepsilon/2$, also $R^{\text{true}}_{p,\phi}(f_r) \geq R^{\text{true}}_{p,\mathbf{1}}(f_r)$ and additionally $R^{\text{true}}_{p,\mathbf{1}}(f_r) \geq \inf_{f\in\mathcal{F}} R^{\text{true}}_{p,\mathbf{1}}(f)$ for every f_r:

$$\begin{aligned}
&\mathbb{P}_{S_+\sim\mathcal{D}^I, S_-\sim\mathcal{D}^K}\left\{\sup_{f\in\mathcal{F}} L(f) \geq \varepsilon\right\} \\
&\leq \sum_{r=1}^{\ell_\varepsilon} 2\exp\left[-2K\max\left\{\frac{\varepsilon^2}{16}\left(R^{\text{true}}_{p,\phi}(f_r)\right)^{2(p-1)}, \left(\frac{\varepsilon}{4}\right)^{2p}\right\}\right] + 2\exp\left[-\frac{\varepsilon^2}{8}I + \ln K\right] \\
&\leq \mathcal{N}\left(\mathcal{F}, \frac{\varepsilon}{8\text{Lip}(\phi)}\right)\left[2\exp\left[-2K\max\left\{\frac{\varepsilon^2}{16}\left(\min_r R^{\text{true}}_{p,\phi}(f_r)\right)^{2(p-1)}, \left(\frac{\varepsilon}{4}\right)^{2p}\right\}\right]\right. \\
&\qquad \left. +2\exp\left[-\frac{\varepsilon^2}{8}I + \ln K\right]\right] \qquad (9) \\
&\leq \mathcal{N}\left(\mathcal{F}, \frac{\varepsilon}{8\text{Lip}(\phi)}\right)\left[2\exp\left[-2K\max\left\{\frac{\varepsilon^2}{16}\left(\inf_{\tilde{f}\in\mathcal{F}} R^{\text{true}}_{p,\mathbf{1}}(\tilde{f})\right)^{2(p-1)}, \left(\frac{\varepsilon}{4}\right)^{2p}\right\}\right]\right. \\
&\qquad \left. +2\exp\left[-\frac{\varepsilon^2}{8}I + \ln K\right]\right].
\end{aligned}$$

Now we put everything together. The probability that there exists an $f \in \mathcal{F}$ where

$$R^{\text{true}}_{p,\phi}(f) \geq R^{\text{empirical}}_{p,\phi}(f) + \varepsilon$$

is at most

$$\mathcal{N}\left(\mathcal{F}, \frac{\varepsilon}{8\text{Lip}(\phi)}\right)\left[2\exp\left[-2K\max\left\{\frac{\varepsilon^2}{16}\left(R_{p,\min}\right)^{2(p-1)}, \left(\frac{\varepsilon}{4}\right)^{2p}\right\}\right] + 2\exp\left[-\frac{\varepsilon^2}{8}I + \ln K\right]\right],$$

where $R_{p,\min} = \inf_f R^{\text{true}}_{p,\mathbf{1}}(f)$. Let us choose $\phi(z) = 1$ for $z \leq 0$, $\phi(z) = 0$ for $z \geq \theta$, and linear in between, with slope $-1/\theta$. Thus, $\text{Lip}(\phi) = 1/\theta$. Since $\phi(z) \leq 1$ for $z \leq \theta$, we have:

$$\begin{aligned} R^{\text{empirical}}_{p,\phi}(f) &= \left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\phi\Big(f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_k)\Big)\right)^p\right)^{1/p} \\ &\leq \left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I}\mathbf{1}_{[f(\mathbf{x}_i)-f(\tilde{\mathbf{x}}_k)\leq\theta]}\right)^p\right)^{1/p} = R^{\text{empirical}}_{p,\mathbf{1},\theta}(f). \end{aligned}$$

Thus, the probability that there exists an $f \in \mathcal{F}$ where

$$R^{\text{true}}_{p,\mathbf{1}}(f) \geq R^{\text{empirical}}_{p,\mathbf{1},\theta}(f) + \varepsilon$$

is at most

$$\mathcal{N}\left(\mathcal{F}, \frac{\varepsilon\theta}{8}\right)\left[2\exp\left[-2K\max\left\{\frac{\varepsilon^2}{16}\left(R_{p,\min}\right)^{2(p-1)}, \left(\frac{\varepsilon}{4}\right)^{2p}\right\}\right] + 2\exp\left[-\frac{\varepsilon^2}{8}I + \ln K\right]\right].$$

Thus, the theorem has been proved. A tighter bound is obtained if we bound differently at (9): instead of using $R^{\text{true}}_{p,\mathbf{1}}(f_r) \geq \inf_{f\in\mathcal{F}} R^{\text{true}}_{p,\mathbf{1}}(f)$, we could stop at $R^{\text{true}}_{p,\mathbf{1}}(f_r) \geq \min_r R^{\text{true}}_{p,\mathbf{1}}(f_r)$ and then choose the $\{f_r\}_r$ to maximize $\min_r R^{\text{true}}_{p,\mathbf{1}}(f_r)$.

Theorem 2 follows directly from the statement of Theorem 3. ■

10.1 1-Dimensional Illustration

As discussed earlier, since most of the value of $R^{\text{true}}_{p,\mathbf{1}}$ comes from a small portion of the domain, more examples are needed to compensate. Let us give a 1-dimensional illustration where this is the case. Almost half the distribution (proportion $\frac{1}{2} - \frac{\varepsilon}{2}$) consists of negative examples uniformly distributed on $[-1,0]$. Almost half the distribution (proportion $\frac{1}{2} - \frac{\varepsilon}{2}$) are positive examples uniformly distributed on $[0,1]$. An $\varepsilon/2$ proportion of the distribution are positive examples distributed on $[-2,-1]$, and the remaining $\varepsilon/2$ are negative examples on $[1,2]$. Drawing a training set of size m from that distribution, with probability $(1-\varepsilon)^m$, all examples will be drawn from $[-1,1]$, missing the essential part of the distribution. Let the hypothesis space $\mathcal{F}$ consist of one monotonically increasing function, and one monotonically decreasing function. Assuming the test set is large and represents the full distribution, the correct function to minimize $R_{\max}$ on the test set is the decreasing function. However, with high probability $(1-\varepsilon)^m$, the increasing function will be (wrongly) chosen, achieving on the training set, $R_{\max} = 0$, but on the test set, the worst possible value $R_{\max} = I$. Thus, $R_{\max}$ relies heavily on an ε-sized portion of the input space. Contrast this with behavior of the AUC, which is hardly affected by this portion of the input space, and is close to 1 with high probability for both training and testing.

11. Proof of Theorem 4

We will use a theorem of Della Pietra et al. (2002), and we will follow their definitions leading to this theorem. Consider a function $\phi : S \subset \mathcal{R}^{IK} \to [-\infty, \infty]$ (unrelated to the ϕ of the proof of Theorem 3). We will use this function to define a Bregman distance and consider an optimization

problem related to this Bregman distance. The dual of this optimization problem will be almost exactly the same as minimization of $R_{p,\exp}$ due to our choice of ϕ. The theorem of Della Pietra et al. (2002) will then provide a kind of uniqueness of the minimizer. The most difficult part of this theorem is finding the function ϕ and showing that the conditions of the framework are satisfied.

Let us first give the definition of a Bregman distance with respect to function ϕ, and then define the primal and dual optimization problems. The *effective domain* of ϕ, denoted Δ_ϕ, is the set of points where ϕ is finite. The function ϕ is *proper* if there is no $\mathbf{p}$ such that $\phi(\mathbf{p}) = -\infty$ and at least some $\mathbf{p}$ with $\phi(\mathbf{p}) \neq \infty$. (Do not confuse the vector $\mathbf{p} \in \mathcal{R}^{IK}$ with the scalar power p. Entries of $\mathbf{p}$ will always be indexed by p_{ik} to avoid confusion.) A proper function ϕ is *essentially smooth* if it is differentiable on the interior of the domain $\text{int}(\Delta_\phi)$ and if $\lim_\ell |\nabla\phi(\mathbf{p}_\ell)| = +\infty$ (element-wise) whenever $\mathbf{p}_\ell$ is a sequence in $\text{int}(\Delta_\phi)$, converging to a point on the boundary. Assume that the function ϕ is *Legendre*, meaning that it is closed (lower semi-continuous), convex and proper, and additionally that $\text{int}(\Delta_\phi)$ is convex, and ϕ is essentially smooth and strictly convex on $\text{int}(\Delta_\phi)$. The *Bregman Distance* associated with ϕ is $B_\phi : \Delta_\phi \times \text{int}(\Delta_\phi) \to [0,\infty]$ defined as:

$$B_\phi(\mathbf{p},\mathbf{q}) := \phi(\mathbf{p}) - \phi(\mathbf{q}) - \langle \nabla\phi(\mathbf{q}), \mathbf{p}-\mathbf{q}\rangle.$$

Fix a vector $\mathbf{p}_0 \in \Delta_\phi$. The *feasible set* for $\mathbf{p}_0$ with respect to matrix $\mathbf{M} \in \mathcal{R}^{IK\times n}$ is: $\mathcal{P} = \{\mathbf{p} \in \mathcal{R}^{IK} | \mathbf{p}^T\mathbf{M} = \mathbf{p}_0^T\mathbf{M}\}$. This will be the domain of the primal problem. The primal problem is to find, for fixed $\mathbf{q}_0 \in \Delta_\phi$:

$$\text{argmin}_{\mathbf{p}\in\mathcal{P}} B_\phi(\mathbf{p},\mathbf{q}_0). \qquad \textit{(primal problem)}$$

Now we lead up to the definition of the dual problem. The *Legendre-Bregman Conjugate* associated with ϕ is $\ell_\phi : \text{int}(\Delta_\phi) \times \mathcal{R}^{IK} \to \mathcal{R} \cup \{\infty\}$ defined as:

$$\ell_\phi(\mathbf{q},\mathbf{v}) := \sup_{\mathbf{p}\in\Delta_\phi} \Big(\langle \mathbf{v},\mathbf{p}\rangle - B_\phi(\mathbf{p},\mathbf{q}) \Big).$$

Note that for fixed $\mathbf{q}$, the Legendre-Bregman conjugate is exactly the convex conjugate of $B_\phi(\cdot,\mathbf{q})$. The *Legendre-Bregman Projection* is the argument of the sup whenever it is well-defined, namely, $\mathcal{L}_\phi : \text{int}(\Delta_\phi) \times \mathcal{R}^{IK} \to \Delta_\phi$ is defined by:

$$\mathcal{L}_\phi(\mathbf{q},\mathbf{v}) := \text{argmax}_{\mathbf{p}\in\Delta_\phi} \Big(\langle \mathbf{v},\mathbf{p}\rangle - B_\phi(\mathbf{p},\mathbf{q}) \Big),$$

whenever this is well-defined. Della Pietra et al. (2002) have shown that:

$$\mathcal{L}_\phi(\mathbf{q},\mathbf{v}) = (\nabla\phi)^{-1}(\nabla\phi(\mathbf{q}) + \mathbf{v}). \tag{10}$$

The dual problem can also be viewed as a minimization of a Bregman distance. Namely, it can be shown (cf. Proposition 2.7 of Della Pietra et al., 2002) that the dual objective can be written in terms of $\mathcal{L}_\phi(\mathbf{q}_0,\mathbf{v})$:

$$\langle \mathbf{v},\mathbf{p}_0\rangle - \ell_\phi(\mathbf{q}_0,\mathbf{v}) = B_\phi(\mathbf{p}_0,\mathbf{q}_0) - B_\phi\Big(\mathbf{p}_0, \mathcal{L}_\phi(\mathbf{q}_0,\mathbf{v})\Big).$$

Thus, since the first term on the right is not a function of $\mathbf{v}$, the dual problem can be written:

$$\begin{aligned} &\text{argmax}_{\mathbf{v}\in\mathcal{R}^{IK}} B_\phi(\mathbf{p}_0,\mathbf{q}_0) - B_\phi\Big(\mathbf{p}_0, \mathcal{L}_\phi(\mathbf{q}_0,\mathbf{v})\Big) \\ = \quad &\text{argmin}_{\mathbf{v}\in\mathcal{R}^{IK}} B_\phi\Big(\mathbf{p}_0, \mathcal{L}_\phi(\mathbf{q}_0,\mathbf{v})\Big), \qquad \textit{(dual problem)} \end{aligned}$$

where we have assumed in the domain of $\mathbf{v}$ that $\Delta_{\phi^*} = \mathcal{R}^{IK}$ and where ϕ^* is the convex conjugate of ϕ. We will rewrite the domain of the dual problem as the class $\mathcal{Q}$, defined as follows. For the $\mathbf{q}_0 \in \Delta_\phi$ and $\mathbf{M} \in \mathcal{R}^{IK \times n}$ fixed in the primal problem, the *Legendre-Bregman projection family for* $\mathbf{q}_0$ *and* $\mathbf{M}$ is defined by:

$$\mathcal{Q}(\mathbf{q}_0, \mathbf{M}) = \{\mathbf{q} \in \Delta_\phi | \mathbf{q} = \mathcal{L}_\phi(\mathbf{q}_0, -\mathbf{M}\boldsymbol{\lambda}) \text{ for some } \boldsymbol{\lambda} \in \mathcal{R}^n\}.$$

So instead of considering the minimizer with respect to $\mathbf{v}$, we will instead consider the minimizer with respect to $\mathbf{q} \in \mathcal{Q}$. In order to proceed, a few more technical conditions are required, namely:

- A1. ϕ is Legendre.
- A2. $\Delta_{\phi^*} = \mathcal{R}^{IK}$ where ϕ^* is the convex conjugate of ϕ.
- A3. B_ϕ extends to a function $B_\phi : \Delta_\phi \times \Delta_\phi \to [0, \infty]$ such that $B_\phi(\mathbf{p}, \mathbf{q})$ is continuous in $\mathbf{p}$ and $\mathbf{q}$, and satisfies $B_\phi(\mathbf{p}, \mathbf{q}) = 0$ iff $\mathbf{p} = \mathbf{q}$.
- A4. $\mathcal{L}_\phi$ extends to a function $\mathcal{L}_\phi : \Delta_\phi \times \mathcal{R}^{IK} \to \Delta_\phi$ satisfying $\mathcal{L}_\phi : (\mathbf{q}, \mathbf{0}) = \mathbf{q}$, such that $\mathcal{L}_\phi(\mathbf{q}, \mathbf{v})$ and $B_\phi(\mathcal{L}_\phi(\mathbf{q}, \mathbf{v}), \mathbf{q})$ are jointly continuous in $\mathbf{q}$ and $\mathbf{v}$.
- A5. $B_\phi(\mathbf{p}, \cdot)$ is *coercive* for every $\mathbf{p} \in \Delta_\phi \backslash \text{int}(\Delta_\phi)$, where a function $f : S \to [-\infty, \infty]$ is coercive if the level sets $\{\mathbf{q} \in S | f(\mathbf{q}) \leq c\}$ are bounded for every $c \in \mathcal{R}$.

We now state Proposition 3.2 of Della Pietra et al. (2002) which will give us uniqueness within the closure of the set $\mathcal{Q}$. Define $\bar{\mathcal{Q}}$ as the closure of $\mathcal{Q}$ in $\mathcal{R}^{IK}$.

Theorem 10 *(Della Pietra et al., 2002) Let* ϕ *satisfy A1.-A5. and suppose that* $\mathbf{p}_0, \mathbf{q}_0 \in \Delta_\phi$ *with* $B_\phi(\mathbf{p}_0, \mathbf{q}_0) < \infty$. *Then there exists a unique* $\mathbf{q}^* \in \Delta_\phi$ *satisfying the following four properties:*

1. $\mathbf{q}^* \in \mathcal{P} \bigcap \bar{\mathcal{Q}}$
2. $B_\phi(\mathbf{p}, \mathbf{q}) = B_\phi(\mathbf{p}, \mathbf{q}^*) + B_\phi(\mathbf{q}^*, \mathbf{q})$ *for any* $\mathbf{p} \in \mathcal{P}$ *and* $\mathbf{q} \in \bar{\mathcal{Q}}$
3. $\mathbf{q}^* = \text{argmin}_{\mathbf{p} \in \mathcal{P}} B_\phi(\mathbf{p}, \mathbf{q}_0)$ *(primal problem)*
4. $\mathbf{q}^* = \text{argmin}_{\mathbf{q} \in \bar{\mathcal{Q}}} B_\phi(\mathbf{p}_0, \mathbf{q})$ *(dual problem)*

Moreover, any one of these four properties determines $\mathbf{q}^*$ *uniquely.*

If we can prove that our objective function fits into this framework, we can use part (4) of this theorem to provide uniqueness in the closure of the set $\mathcal{Q}$, which will be related to our set $\mathcal{Q}'$. Let us now do exactly this.

Consider the following function $\phi : \mathcal{R}^{IK}_{\geq 0} \to [-\infty, \infty]$:

$$\phi(\mathbf{q}) := \sum_{ik} q_{ik} \gamma(q_{ik}, \mathbf{q}), \text{ where } \gamma(q_{ik}, \mathbf{q}) := \ln \left(\frac{q_{ik}}{p^{1/p} (\sum_{i'} q_{i'k})^{(p-1)/p}} \right).$$

We extend the definition to $\mathcal{R}^{IK}_+$ by the conventions $0 \ln 0 = 0$ and $q_{ik}\gamma(q_{ik}, \mathbf{q}) = 0$ whenever $q_{ik} = 0$ for all i. Thus, Δ_ϕ is now $\mathcal{R}^{IK}_+$. The boundary in our case is where q_{ik} equals 0 for one or more ik pairs. We must now show that ϕ is Legendre.

Lemma 11 *ϕ is strictly convex in* $\mathrm{int}(\Delta_\phi)$*, where Δ_ϕ are vectors in $\mathcal{R}_+^{IK}$ with strictly positive entries.*

The proof is in the Appendix.

Lemma 12 *ϕ is Legendre.*

Proof ϕ is proper since there is no $\mathbf{q}$ such that $\phi(\mathbf{q}) = -\infty$. In order to achieve this, the term inside the logarithm must be exactly zero. When that happens, $q_{ik} = 0$, and by our convention, $q_{ik}\gamma(q_{ik}, \mathbf{q}) = 0$, thus the entire ik term is zero rather than $-\infty$. It can be verified that ϕ is lower semi-continuous. Also, $\mathrm{int}(\Delta_\phi) = \mathcal{R}_{\gg 0}^{IK}$ which is convex. We have already shown that ϕ is strictly convex on $\mathrm{int}(\Delta_\phi)$ in Lemma 11, and by our definition of ϕ on the boundary, it is convex on Δ_ϕ. We now show that ϕ is essentially smooth with respect to the boundary. Consider the following calculation for the gradient of ϕ in $\mathrm{int}(\Delta_\phi)$:

$$(\nabla\phi(\mathbf{q}))_{ik} = \frac{\partial\phi(\mathbf{q})}{\partial q_{ik}} = \frac{1}{p} + \ln\left(\frac{q_{ik}}{p^{1/p}\left(\sum_{i'} q_{i'k}\right)^{(p-1)/p}}\right) = \frac{1}{p} + \gamma(q_{ik}, \mathbf{q}), \tag{11}$$

since $\gamma(q_{ik}, \mathbf{q}) \to -\infty$ as $q_{ik} \to 0$, ϕ is essentially smooth. All the conditions have now been checked. ■

Also, we require the following:

Lemma 13 *Conditions A1.-A5. are obeyed.*

The proof of this lemma is in the Appendix.

Proof *(Of Theorem 4)* Let us compute the quantities above for our function ϕ, namely we would like to find the space Q and the dual objective $B_\phi(\mathbf{p}_0, \mathbf{q})$. Using (11) it can be shown that:

$$\left((\nabla\phi)^{-1}(\mathbf{z})\right)_{ik} = pe^{(z_{ik}-1/p)}\left(\sum_{i'} e^{(z_{i'k}-1/p)}\right)^{p-1}.$$

We now wish to compute $\mathcal{L}_\phi$. First, let us compute a term that appears often:

$e^{z_{ik}-1/p}$ where $z_{ik} = (\nabla\phi(\mathbf{q}) + \mathbf{v})_{ik} = \dfrac{1}{p} + \gamma(q_{ik}, \mathbf{q}) + v_{ik}$ can be rewritten:

$$e^{z_{ik}-1/p} \quad = \quad \exp\left[\frac{1}{p} - \frac{1}{p} + \gamma(q_{ik}, \mathbf{q}) + v_{ik}\right] = e^{v_{ik}}e^{\gamma(q_{ik}, \mathbf{q})}.$$

Thus from (10),

$$\begin{aligned}
\mathcal{L}_\phi(\mathbf{q}, \mathbf{v})_{ik} &= pe^{v_{ik}}e^{\gamma(q_{ik}, \mathbf{q})}\left(\sum_{i'} e^{v_{i'k}}e^{\gamma(q_{i'k}, \mathbf{q})}\right)^{p-1} \\
&= pe^{v_{ik}}\frac{q_{ik}}{p^{1/p}(\sum_{i'} q_{i'k})^{(p-1)/p}}\left(\frac{\sum_{i'} e^{v_{i'k}}q_{i'k}}{p^{1/p}(\sum_{i'} q_{i'k})^{(p-1)/p}}\right)^{p-1} \\
&= e^{v_{ik}}q_{ik}\left(\sum_{i'} e^{v_{i'k}}q_{i'k}\right)^{(p-1)}\frac{1}{(\sum_{i'} q_{i'k})^{(p-1)}}.
\end{aligned} \tag{12}$$

In our case, we choose $\mathbf{q}_0$ to be constant, $q_{0ik} = q_0$ for all i,k. We can now obtain $\mathcal{Q}$:

$$\mathcal{Q}(\mathbf{q}_0, \mathbf{M}) = \left\{ \mathbf{q} \Big| \mathbf{q} = e^{-(\mathbf{M}\boldsymbol{\lambda})_{ik}} \left(\sum_{i'} e^{-(\mathbf{M}\boldsymbol{\lambda})_{i'k}} \right)^{(p-1)} \frac{q_0}{I^{(p-1)}} \text{ for some } \boldsymbol{\lambda} \in \mathcal{R}^n \right\}.$$

In order to make the last fraction become 1, we choose $q_0 = I^{(p-1)}$. We now need only to define $\mathbf{p}_0$ in order to define the dual problem. In our case, we choose $\mathbf{p}_0 = \mathbf{0}$ so that the dual objective function is $B_\phi(\mathbf{0}, \mathbf{q})$. Let us choose $\mathbf{q} \in \mathcal{Q}$, that is, $\mathbf{q}_{ik} = e^{-(\mathbf{M}\boldsymbol{\lambda})_{ik}} \left(\sum_{i'} e^{-(\mathbf{M}\boldsymbol{\lambda})_{i'k}} \right)^{(p-1)}$ and substitute using (11) and the definitions of ϕ and B_ϕ:

$$\begin{aligned}
B_\phi(\mathbf{0}, \mathbf{q}) &= \phi(\mathbf{0}) - \phi(\mathbf{q}) - \langle \nabla\phi(\mathbf{q}), \mathbf{0} - \mathbf{q} \rangle \\
&= -\phi(\mathbf{q}) + \mathbf{q} \cdot \nabla\phi(\mathbf{q}) \\
&= -\phi(\mathbf{q}) + \frac{1}{p} \sum_{ik} q_{ik} + \phi(\mathbf{q}) \\
&= \frac{1}{p} \sum_k \left(\sum_i e^{-(\mathbf{M}\boldsymbol{\lambda})_{ik}} \right) \left(\sum_{i'} e^{-(\mathbf{M}\boldsymbol{\lambda})_{i'k}} \right)^{(p-1)} \\
&= \frac{1}{p} \sum_k \left(\sum_i e^{-(\mathbf{M}\boldsymbol{\lambda})_{ik}} \right)^p = \frac{1}{p} R_{p,\exp}(\boldsymbol{\lambda}).
\end{aligned}$$

Thus, we have arrived at exactly the objective function for our algorithm. In other words, the function ϕ was carefully chosen so that the dual objective would be exactly as we wished, modulo the constant factor $1/p$ which does not affect minimization.

Part 4 of Theorem 10 tells us that the objective function of our algorithm has a unique minimizer in $\bar{\mathcal{Q}}$ as long as A1.-A5. are obeyed, which holds from Lemma 13. It remains only to show that a vector in $\bar{\mathcal{Q}}$ yields a unique vector in $\bar{\mathcal{Q}}'$. Consider a sequence of vectors in $\mathcal{Q}$ defined element-wise by $q_{\ell,ik} = e^{-(\mathbf{M}\boldsymbol{\lambda})_{\ell,ik}} \left(\sum_{i'} e^{-(\mathbf{M}\boldsymbol{\lambda})_{\ell,i'k}} \right)^{p-1}$ such that $q_\ell \to \bar{q}$ as $\ell \to \infty$. Then consider the sequence defined by:

$$\frac{q_{\ell,ik}}{\left(\sum_{i'} q_{\ell,i'k}\right)^{(p-1)/p}} = e^{-(\mathbf{M}\boldsymbol{\lambda})_{\ell,ik}}.$$

By definition of $\mathcal{Q}'$, each vector in this sequence is in $\mathcal{Q}'$. This sequence converges pointwise to $\frac{\bar{q}_{ik}}{\left(\sum_{i'} \bar{q}_{i'k}\right)^{(p-1)/p}} \in \bar{\mathcal{Q}}'$, or if $\bar{q}_{ik} = 0$, then the ik^{th} component of the sequence converges to 0. Since we are in a finite dimensional space, namely $\mathcal{R}^{IK}$, pointwise convergence is sufficient. ■

It was unnecessary to state the primary objective $B_\phi(\mathbf{p}, \mathbf{q}_0)$ explicitly to prove the theorem, however, we state it in order to compare with the relative entropy case where $p = 1$. Recall that $\mathbf{q}_0$ is the constant vector with entries I^{p-1}. Thus, $(\nabla\phi(\mathbf{q}_0))_{ik} = \frac{1}{p} + \gamma(q_0, \mathbf{q}_0) = \frac{1}{p} + \ln(q_0 / [p^{1/p} (I q_0)^{(p-1)/p}]) = \frac{1}{p}(1 - \ln p)$ for all ik.

$$\begin{aligned}
B_\phi(\mathbf{p}, \mathbf{q}_0) &= \phi(\mathbf{p}) - \phi(\mathbf{q}_0) - \langle \nabla\phi(\mathbf{q}_0), \mathbf{p} - \mathbf{q}_0 \rangle \\
&= \phi(\mathbf{p}) - \langle \nabla\phi(\mathbf{q}_0), \mathbf{p} \rangle + \frac{1}{p} I^p K = \phi(\mathbf{p}) - (\nabla\phi(\mathbf{q}_0))_{ik} \sum_{ik} p_{ik} + \frac{1}{p} I^p K \\
&= \sum_{ik} p_{ik} \ln \left[\frac{p_{ik}}{p^{1/p} \left(\sum_{i'} p_{i'k}\right)^{(p-1)/p}} \right] - \frac{1}{p}(1 - \ln p) \sum_{ik} p_{ik} + \frac{1}{p} I^p K.
\end{aligned}$$

For $p = 1$ this reduces exactly to the relative entropy case.

One interesting note is how to find a function ϕ to suit such a problem; when we introduced it, we gave no indication of the techniques required to find such a function. In this case, we discovered the function ϕ again via convex duality. We knew the desired dual problem was precisely our objective $R_{p,\exp}$, thus, we were able to recover the primal problem by convex conjugation. The double dual in this case is the objective itself. From there, the function ϕ was obtained by analogy with the relative entropy case.

12. Conclusions

We have provided a method for constructing a ranked list where correctness at the top of the list is most important. Our main contribution is a general set of convex objective functions determined by a loss ℓ and price function g. A boosting-style algorithm based on a specific family of these objectives is derived. We have demonstrated the effect of a number of different price functions, and it is clear, both theoretically and empirically, that a steeper price function concentrates harder at the top of the list.

Acknowledgments

Thanks to the anonymous reviewers regarding Theorem 3, the experiments, and parts of the discussion, and especially for finding the numerous errors that come about from single authorship. Thanks to Rob Schapire for his advice, particularly with Lemma 11, thanks to Adrian Banner for careful proofreading, and thanks to Sinan Güntürk, particularly regarding Lemma 7. Also thanks to Eero Simoncelli and Dave Waltz. This work was supported by an NSF postdoctoral research fellowship under grant DBI-0434636 at New York University.

Appendix A.

We provide proofs of Lemma 11 and Lemma 13.

Proof (Of Lemma 11) First, rewrite ϕ:

$$\phi(\mathbf{q}) = \sum_k \left[\left(\sum_i q_{ik} \ln q_{ik} \right) - (\ln p^{1/p}) \left(\sum_i q_{ik} \right) - \frac{p-1}{p} \left(\sum_i q_{ik} \right) \ln \left(\sum_i q_{ik} \right) \right].$$

The middle term is linear so it does not affect convexity of the sum. It is sufficient to prove convexity of the following function, since ϕ would then be a sum (over k) of convex functions. Define $f : \mathcal{R}^I_{\geqslant 0} \to \mathcal{R}$ as follows, for $\mathbf{q} \in \mathcal{R}^I_+$:

$$f(\mathbf{q}) := \left(\sum_i q_i \ln q_i \right) + \frac{1-p}{p} \left(\sum_i q_i \right) \ln \left(\sum_i q_i \right).$$

Thus, the Hessian is:

$$\frac{\partial f(\mathbf{q})}{\partial q_\ell \partial q_i} = \frac{1}{q_i} \delta_{i=\ell} + \frac{1-p}{p} \frac{1}{\sum_{i'} q_{i'}}.$$

To show that the Hessian is positive definite, we show that $\mathbf{w}^{\mathbf{T}}\mathbf{H}\mathbf{w} > 0$ whenever $\mathbf{w} \neq \mathbf{0}$.

$$\begin{aligned}\sum_{i\ell} w_i w_\ell \frac{\partial f}{\partial q_\ell \partial q_i} &= \sum_i w_i^2 \frac{1}{q_i} + \frac{1-p}{p}\left(\sum_i w_i\right)^2 \frac{1}{\sum_i q_i} \\ &= \left(\frac{1}{\sum_i q_i}\right)\left[\left(\sum_i w_i^2 \frac{1}{q_i}\right)\left(\sum_i q_i\right) + \left(\frac{1}{p}-1\right)\left(\sum_i w_i\right)^2\right].\end{aligned}$$

Now, consider the Cauchy-Schwarz inequality, used in the following way:

$$\left(\sum_i w_i\right)^2 = \left\langle \frac{\mathbf{w}}{\sqrt{\mathbf{q}}}, \sqrt{\mathbf{q}}\right\rangle^2 \leq \left\|\frac{\mathbf{w}}{\sqrt{\mathbf{q}}}\right\|_2^2 \|\sqrt{\mathbf{q}}\|_2^2 = \left(\sum_i \frac{w_i^2}{q_i}\right)\left(\sum_i q_i\right).$$

Substituting back,

$$\begin{aligned}\sum_{i\ell} w_i w_\ell \frac{\partial f}{\partial q_\ell \partial q_i} &\geq \left(\frac{1}{\sum_i q_i}\right)\left[\left(\sum_i w_i^2 \frac{1}{q_i}\right)\left(\sum_i q_i\right) + \frac{1}{p}\left(\sum_i w_i\right)^2 - \left(\sum_i w_i^2 \frac{1}{q_i}\right)\left(\sum_i q_i\right)\right] \\ &= \left(\frac{1}{\sum_i q_i}\right)\frac{1}{p}\left(\sum_i w_i\right)^2.\end{aligned}$$

Recall that equality in Cauchy-Schwarz is only achieved when vectors are dependent, that is, for some $\alpha \in \mathcal{R}$, $w_i = \alpha q_i$ for all i. Since the elements of $\mathbf{q}$ are all strictly positive, if $w_i = \alpha q_i$, then at the same time we cannot have $\sum_i w_i = 0$. Thus, when equality holds in Cauchy-Schwarz, then $(\sum_i w_i)^2 > 0$ unless $\mathbf{w} = \mathbf{0}$. Thus, whether the Cauchy-Schwarz inequality is strict or not, we have:

$$\sum_{i\ell} w_i w_\ell \frac{\partial f}{\partial q_\ell \partial q_i} > 0 \text{ whenever } \mathbf{w} \neq \mathbf{0}.$$

Thus, ϕ is strictly convex. ■

Proof *(Of Lemma 13)* Condition A1. was proven in Lemma 12. To show A2., note that:

$$(\phi^*(\mathbf{v}))_{ik} = p e^{(v_{ik}-1/p)}\left(\sum_{i'} e^{(v_{i'k}-1/p)}\right)^{p-1}.$$

Thus, $\Delta_{\phi^*} = \mathcal{R}^{IK}$.

For A3., let us simplify using (11), where this calculation is valid for $\mathbf{p}, \mathbf{q} \in \Delta_\phi \times \text{int}(\Delta_\phi)$:

$$\begin{aligned}B_\phi(\mathbf{p},\mathbf{q}) &= \phi(\mathbf{p}) - \phi(\mathbf{q}) - \nabla\phi(\mathbf{q})\cdot(\mathbf{p}-\mathbf{q}) \\ &= \sum_{ik} p_{ik}\gamma(p_{ik},\mathbf{p}) - \phi(\mathbf{q}) - \frac{1}{p}\sum_{ki}(p_{ik}-q_{ik}) - \sum_{ik} p_{ik}\gamma(q_{ik},\mathbf{q}) + \phi(\mathbf{q}) \\ &= \sum_{ik} p_{ik}\Big(\gamma(p_{ik},\mathbf{p}) - \gamma(q_{ik},\mathbf{q})\Big) - \frac{1}{p}\sum_{ik}(p_{ik}-q_{ik}). \qquad (13)\end{aligned}$$

Now we consider the boundary. If for some ik pair, $p_{ik}=0$ then let $p_{ik}(\gamma(p_{ik},\mathbf{p})-\gamma(q_{ik},\mathbf{q}))=0$ for all $\mathbf{q}$. If for some ik pair, $p_{ik}\neq 0$ and additionally $q_{ik}=0$ we define $B_\phi(\mathbf{p},\mathbf{q})=\infty$. This completes our definition of B_ϕ on the boundary of $\mathcal{R}_+^{IK}$. Let us prove that $B_\phi(\mathbf{p},\mathbf{q})=0$ implies $\mathbf{p}=\mathbf{q}$. Considering the interior, B_ϕ can only be 0 at a minimum since it is non-negative. A necessary condition for B_ϕ to be at a minimum is for $\partial B_\phi(\mathbf{p},\mathbf{q})/\partial p_{ik}=0$ for all ik:

$$\forall ik \quad 0=\frac{\partial B_\phi(\mathbf{p},\mathbf{q})}{\partial p_{ik}}=1-\frac{p-1}{p}+\gamma(p_{ik},\mathbf{p})-\gamma(q_{ik},\mathbf{q})-\frac{1}{p}\Rightarrow \forall ik \quad \gamma(p_{ik},\mathbf{p})-\gamma(q_{ik},\mathbf{q})=0.$$

It is true that $\gamma(p_{ik},\mathbf{p})-\gamma(q_{ik},\mathbf{q})=0$ for pair ik implies that $p_{ik}=q_{ik}$. To see this, note that one can determine p_{ik} directly from the $\gamma(p_{ik},\mathbf{p})$ values as follows. Set $z_{ik}:=p^{1/p}\exp(\gamma(p_{ik},\mathbf{p}))$. Now,

$$z_{ik}\left(\sum_{i'} z_{i'k}\right)^{p-1}=p_{ik}.$$

Hence, $\forall ik, \gamma(p_{ik},\mathbf{p})-\gamma(q_{ik},\mathbf{q})=0$ implies that $\mathbf{p}=\mathbf{q}$. Consider now the boundary. If for any ik, $p_{ik}\neq 0$ and $q_{ik}=0$ then $B_\phi(\mathbf{p},\mathbf{q})=\infty\neq 0$. So, if $B_\phi(\mathbf{p},\mathbf{q})=0$, then whenever $q_{ik}=0$ we must have $p_{ik}=0$. On the other hand, if $p_{ik}=0$, there will be a contribution to $B_\phi(\mathbf{p},\mathbf{q})$ of $\frac{1}{p}q_{ik}$, implying that q_{ik} must be 0 in order for $B_\phi(\mathbf{p},\mathbf{q})=0$. Thus, A3. holds.

We now show A4. Let us define the boundary values for $\mathcal{L}_\phi$. If for some k we have $\sum_{i'} q_{i'k}=0$, then let $(\mathcal{L}_\phi(\mathbf{q},\mathbf{v}))_{ik}=0$ for all i. Otherwise, (12) can be used as written. Thus, we always have $\mathcal{L}_\phi(\mathbf{q},\mathbf{0})=\mathbf{q}$, and $\mathcal{L}_\phi(\mathbf{q},\mathbf{v})$ is jointly continuous in $\mathbf{q}$ and $\mathbf{v}$. Now consider $B_\phi(\mathcal{L}_\phi(\mathbf{q},\mathbf{v}),\mathbf{q})$. Let us simplify this expression in the interior, starting from (12) and (13) and using the notation $\{\mathcal{L}\}_{ik}$ for the vector $\{\mathcal{L}_\phi(\mathbf{q},\mathbf{v})\}_{ik}$.

$$\begin{aligned}
B_\phi(\mathcal{L},\mathbf{q}) &= \sum_{ik}\mathcal{L}_{ik}\Big(\gamma(\mathcal{L}_{ik},\mathcal{L})-\gamma(q_{ik},\mathbf{q})\Big)-\frac{1}{p}\sum_{ik}(\mathcal{L}_{ik}-q_{ik})\\
&= \sum_{ik}\mathcal{L}_{ik}\ln\left(\frac{\mathcal{L}_{ik}}{p^{1/p}(\sum_{i'}\mathcal{L}_{i'k})^{(p-1)/p}}\frac{p^{1/p}(\sum_{i'}q_{i'k})^{(p-1)/p}}{q_{ik}}\right)-\frac{1}{p}\sum_{ik}(\mathcal{L}_{ik}-q_{ik})\\
&= \sum_{ik}\mathcal{L}_{ik}\ln\left[\frac{e^{v_{ik}}q_{ik}(\sum_{i'}e^{v_{i'k}}q_{i'k})^{p-1}\left(\frac{1}{\sum_{i'}q_{i'k}}\right)^{p-1}}{\left[(\sum_{i'}e^{v_{i'k}}q_{i'k})^p\left(\frac{1}{\sum_i q_{i'k}}\right)^{p-1}\right]^{(p-1)/p}}\frac{(\sum_{i'}q_{i'k})^{(p-1)/p}}{q_{ik}}\right]\\
&\qquad -\frac{1}{p}\sum_{ik}(\mathcal{L}_{ik}-q_{ik})\\
&= \sum_{ik}\mathcal{L}_{ik}v_{ik}-\frac{1}{p}\sum_{ik}(\mathcal{L}_{ik}-q_{ik}).
\end{aligned}$$

Thus, since $\mathcal{L}_\phi$ is jointly continuous in $\mathbf{q}$ and $\mathbf{v}$, B_ϕ is jointly continuous in $\mathbf{q}$ and $\mathbf{v}$.

For A5., we need to show that $B_\phi(\mathbf{p},\cdot)$ is coercive, meaning that the level set $\{\mathbf{q}\in\Delta_\phi : B_\phi(\mathbf{p},\mathbf{q})\leq c\}$ is bounded, with $\mathbf{p}\in\Delta_\phi\backslash\text{int}(\Delta_\phi)$ which are vectors in $\mathcal{R}_+^{IK}$ with at least one entry that is 0. Recall that we use the convention $0\ln 0=0$. Consider from (13), using the fact that for any ik pair,

$\ln q_{ik}^{(p-1)/p} \leq \ln \left(\sum_i q_{ik}\right)^{(p-1)/p}$:

$$\begin{aligned} B_\phi(\mathbf{p},\mathbf{q}) &= \sum_{ik} p_{ik}\Big(\gamma(p_{ik},\mathbf{p}) - \gamma(q_{ik},\mathbf{q})\Big) - \frac{1}{p}\sum_{ik}(p_{ik}-q_{ik}) \\ &= \sum_{ik} -p_{ik}\gamma(q_{ik},\mathbf{q}) + \frac{1}{p}q_{ik} + \text{function}(\mathbf{p}) \\ &\geq \sum_{ik} -p_{ik}\ln q_{ik} + p_{ik}\ln\left(q_{ik}^{(p-1)/p}\right) + \frac{1}{p}q_{ik} + \text{function}(\mathbf{p}) \\ &= \frac{1}{p}\sum_{ik} -p_{ik}\ln q_{ik} + q_{ik} + \text{function}(\mathbf{p}). \end{aligned}$$

Since logarithms grow slowly, one can choose a q_{ik} large enough so that this sum exceeds any fixed constant c, regardless of the values of the other q_{ik}'s. Thus, the set $\{\mathbf{q} \in \Delta_\phi : B_\phi(\mathbf{p},\mathbf{q}) \leq c\}$ is bounded. We are done checking the conditions. ■

References

Shivani Agarwal, Thore Graepel, Ralf Herbich, Sariel Har-Peled, and Dan Roth. Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research*, 6:393–425, 2005.

Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Olivier Bousquet. New approaches to statistical learning theory. *Annals of the Institute of Statistical Mathematics*, 55(2):371–389, 2003.

Ulf Brefeld and Tobias Scheffer. AUC maximizing support vector learning. In *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning*, 2005.

Stéphan Clemençon and Nicolas Vayatis. Ranking the best instances. *Journal of Machine Learning Research*, 8:2671–2699, Dec 2007.

Stéphan Clemençon and Nicolas Vayatis. Empirical performance maximization for linear rank statistics. In *Advances in Neural Information Processing Systems 22*, 2008.

Stéphan Clemençon, Gabor Lugosi, and Nicolas Vayatis. Ranking and empirical minimization of U-statistics. *The Annals of Statistics*, 36(2):844–874, 2008.

Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1/2/3), 2002.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

David Cossock and Tong Zhang. Subset ranking using regression. In *Proceedings of the Ninteenth Annual Conference on Learning Theory*, 2006.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc. (N.S.)*, 39(1):1–49, 2002.

Ofer Dekel, Christopher Manning, and Yoram Singer. Log-linear models for label ranking. In *Advances in Neural Information Processing Systems 16*, 2004.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Duality and auxiliary functions for Bregman distances. Technical Report CMU-CS-01-109R, School of Computer Science, Carnegie Mellon University, 2002.

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.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

Simon I. Hill, Hugo Zaragoza T, Ralf Herbrich T, and Peter J. W. Rayner. Average precision and the problem of generalisation. In *In Proceedings of the ACM SIGIR Workshop on Mathematical and Formal Methods in Information Retrieval*, 2002.

Kalervo Järvelin and Jaana Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR '00: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48, New York, NY, USA, 2000. ACM.

Heng Ji, Cynthia Rudin, and Ralph Grishman. Re-ranking algorithms for name tagging. In *HLT/NAACL workshop on computationally hard problems and joint interference in speech and language processing*, 2006.

Vladimir Koltchinskii and Dmitry Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1), February 2002.

Quoc Le and Alex Smola. Direct optimization of ranking measures. arXiv:0704.3359v1, November 2007.

Sofus A. Macskassy, Foster Provost, and Saharon Rosset. Pointwise ROC confidence bounds: An empirical evaluation. In *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning*, 2005.

Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics 1989*, pages 148–188. Cambridge University Press, 1989.

Michael C. Mozer, Robert Dodier, Michael D. Colagrosso, Csar Guerra-salcedo, and Richard Wolniewicz. Prodding the ROC curve: Constrained optimization of classifier performance. In *Advances in Neural Information Processing Systems 14*, pages 1409–1415, 2002.

Alain Rakotomamonjy. Optimizing AUC with support vector machine (SVM). In *Proceedings of European Conference on Artificial Intelligence Workshop on ROC Curve and AI, Valencia, Spain*, 2004.

Cynthia Rudin. Ranking with a p-norm push. In *Proceedings of the Ninteenth Annual Conference on Learning Theory*, pages 589–604, 2006.

Cynthia Rudin and Robert E. Schapire. Margin-based ranking and an equivalence between AdaBoost and RankBoost. *Journal of Machine Learning Research*, 10:2193–2232, October 2009.

Cynthia Rudin, Corinna Cortes, Mehryar Mohri, and Robert E. Schapire. Margin-based ranking meets boosting in the middle. In Peter Auer and Ron Meir, editors, *Proceedings of the Eighteenth Annual Conference on Learning Theory*, pages 63–78. Springer, 2005.

Cynthia Rudin, Rebecca Passonneau, Axinia Radeva, Haimonti Dutta, Steve Ierome, and Delfina Isaac. Predicting manhole events in Manhattan : A case study in extended knowledge discovery. Accepted for publication to Machine Learning, 2009.

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.

Shai Shalev-Shwartz and Yoram Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 7:1567–1599, December 2006.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, Sept 2005.

Nicolas Usunier, Massih-Reza Amini, and Patrick Gallinari. A data-dependent generalisation error bound for the AUC. In *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning*, 2005.

Lian Yan, Robert H. Dodier, Michael Mozer, and Richard H. Wolniewicz. Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 848–855, 2003.

Tong Zhang and Bin Yu. Boosting with early stopping - convergence and consistency. *The Annals of Statistics*, 33(4):1538–1579, 2005.

Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in Neural Information Processing Systems 19*, 2007.

Journal of Machine Learning Research 10 (2009) 2273-2293 Submitted 11/08; Revised 2/09; Published 10/09

Learning Nondeterministic Classifiers

Juan José del Coz JUANJO@AIC.UNIOVI.ES
Jorge Díez JDIEZ@AIC.UNIOVI.ES
Antonio Bahamonde ANTONIO@AIC.UNIOVI.ES
Artificial Intelligence Center
University of Oviedo at Gijón
Asturias, Spain

Editor: Lyle Ungar

Abstract

Nondeterministic classifiers are defined as those allowed to predict more than one class for some entries from an input space. Given that the true class should be included in predictions and the number of classes predicted should be as small as possible, these kind of classifiers can be considered as Information Retrieval (IR) procedures. In this paper, we propose a family of IR loss functions to measure the performance of nondeterministic learners. After discussing such measures, we derive an algorithm for learning optimal nondeterministic hypotheses. Given an entry from the input space, the algorithm requires the posterior probabilities to compute the subset of classes with the lowest expected loss. From a general point of view, nondeterministic classifiers provide an improvement in the proportion of predictions that include the true class compared to their deterministic counterparts; the price to be paid for this increase is usually a tiny proportion of predictions with more than one class. The paper includes an extensive experimental study using three deterministic learners to estimate posterior probabilities: a multiclass Support Vector Machine (SVM), a Logistic Regression, and a Naïve Bayes. The data sets considered comprise both UCI multi-class learning tasks and microarray expressions of different kinds of cancer. We successfully compare nondeterministic classifiers with other alternative approaches. Additionally, we shall see how the quality of posterior probabilities (measured by the Brier score) determines the goodness of nondeterministic predictions.

Keywords: nondeterministic, multiclassification, reject option, multi-label classification, posterior probabilities

1. Introduction

There are several learners that successfully solve classification tasks in which the number of classes is higher than two; see for instance Wu et al. (2004) and Lin et al. (2008). However, for each class C most classification errors frequently occur between small subsets of classes that are somehow similar to C, regardless of the approach used. This fact suggests that multiclass classifiers would increase in reliability if they were allowed to express their doubts whenever they were asked to classify some entries.

In this paper we explore how to learn classifiers with multiple outcomes, like nondeterministic automata; we shall call them *nondeterministic classifiers*. Since they return a set of values, these classifiers could be called *set-valued* classifiers. To fix ideas, let us consider a screening for a set of medical diseases (or other diagnostic situations); for some inputs, a nondeterministic classifier

would be able to predict not just one single disease, but a set of options. These multiple predictions will be provided to domain experts when the classifier is not sure enough to give a unique class. Thus nondeterministic predictions may discard some options and allow domain experts to make practical decisions. Even when the nondeterministic classifier returns most of the available classes for the representation of an entry, we can read that the learned hypothesis is acknowledging its ignorance about how to deal with that entry.

It is evident that nondeterministic classifiers will include *true* classes in their predictions more frequently than deterministic hypotheses: they only have one possibility to be right. In this sense, nondeterministic predictions are backed by greater reliability. To be useful, however, nondeterministic classifiers should not only predict a set of classes containing the correct or true one, but their prediction sets should also be as small as possible. Notice that these requirements are common in algorithms designed for *Information Retrieval*. In this case, the queries are the entries to be classified and the *Recall* and *Precision* are then applied to each prediction. Hence, the loss functions for nondeterministic classifiers can be built as combinations of IR measures, as F_β functions are.

Starting from the distribution of posterior probabilities of classes, given one entry, we present an algorithm that computes the subset of classes with the lowest expected loss. In the experiments reported at the end of the paper, we employed three deterministic learners that provide posterior probabilities: Support Vector Machines *(SVM)*, Logistic Regression *(LR)*, and Naïve Bayes *(NB)*. We successfully compared the achievements of our nondeterministic classifiers with those obtained by other alternative approaches.

The paper is organized as follows. In the next section, we present an overview of related work on classifiers that return subsets of classes instead of a single class. The formal settings both for nondeterministic classifiers and their loss functions are presented in the third section. After that, in Section 4, we derive an algorithm to learn nondeterministic hypotheses. Then, we conclude the paper with a section in which we report an experimental study of their performance. In addition to the comparison mentioned above, we discuss the role played by the deterministic learner that provides posterior probabilities. We see that the quality of posterior probabilities determines the goodness of nondeterministic predictions. The data sets used are publicly available and, in addition to a group of data sets from the UCI Repository (Asuncion and Newman, 2007), they include a group of classification tasks of cancer patients from gene expressions captured by microarrays.

2. Related Work

Nondeterministic classifiers are somehow related to classifiers with *reject option* (Chow, 1970). In this approach, the entries that are likely to be misclassified are rejected, they are not classified and can be handled by more sophisticated procedures: a manual classification, for instance. The core assumption is that the cost of making a wrong decision is 1, while the cost of using the reject option is given by some $d, 0 < d < 1$. In this context, provided that posterior probabilities are exactly known, an optimal rejection rule can be devised (Chow, 1970; Bartlett and Wegkamp, 2008): an entry is rejected if the maximum posterior probability is less than a threshold. Notice that classifiers with reject option are a relaxed version of nondeterministic classifiers. Rejection is a nondeterministic classification that includes the complete set of classes. On the other hand, instead of avoiding difficult classifications, for each entry, nondeterministic classifers adventure a set of possible classes, not necessarily the complete set.

However, predictors of more than one class are not completely new. Given an $\varepsilon \in [0,1]$, the so-called confidence machines make *conformal predictions* (Shafer and Vovk, 2008): they produce a set of labels containing the true class with a probability greater than $1-\varepsilon$.

To the best of our knowledge, the most directly related work to the approach presented in this paper is that of Zaffalon (2002) and Corani and Zaffalon (2008a,b). In these papers, the authors describe the *Naïve Credal Classifier*, a set-valued classifier which is an extension of the Naïve Bayes classifier to imprecise probabilities. The Naïve Credal Classifier models prior ignorance about the distribution of classes by means of a set of prior densities (also called the prior *credal set*), which is turned into a set of posterior probabilities by element-wise application of Bayes' rule. The classifier returns all the classes that are *non-dominated* by any other class according to the posterior credal set.

Another learning task that is related to this paper is multi-label classification. However, training instances in multi-label tasks can belong to more than one class, while nondeterministic training sets are the same as those of standard classification. In Tsoumakas and Katakis (2007), the authors provide an in-depth description of multi-label classification, enumerate several methods and compare their performance using Information Retrieval measures. Some applications have likewise arisen within the context of hierarchical organization of biological objects: predicting gene functions (Clare and King, 2003), or mapping biological entities into ontologies (Kriegel et al., 2004).

The formal setting presented in this paper was previously introduced in Alonso et al. (2008). There, we dealt with an interesting application of nondeterministic classifiers, in which classes (or *ranks*, in that context) are linearly ordered. The aim was to predict the rank (in an ordered scale) of carcasses of beef cattle. This value determines, on the one hand, the prices to be obtained by carcasses and, on the other, the genetic value of animals in order to select studs for the next generation. In this application, nondeterministic classifiers return an interval of ranks. Interval predictions are useful even when the intervals comprise more than one rank. For instance, it is possible to reject an animal as a stud for the next generation when a prediction interval is included in the lowest part of the scale. However, if we need a unique rank, we may decide to appeal to an *actual* expert to resolve the ambiguity, an expensive classification procedure not always available in practice.

The novelty of this paper is that now we deal with a standard classification setting; that is, the sets of classes are not ordered. This fact is very important as the search for the optimal prediction leads to a dramatic difference in complexity. Thus, if k is the number of classes, the search in the ordinal case is just of order k^2, while in the unordered case, at a first glance, the search is of order 2^k. However, the Theorem of Correctness of Algorithm 1 proves that this search can be accomplished in polynomial time.

Additionally, this paper reports an extensive experimental study. First, we test whether nondeterministic classifiers outperform *Naïve Credal Classifiers* and other alternative approaches. We then investigate the role played by the ingredients of nondeterministic classifiers.

3. Formal Presentation and Notation

Let $\mathcal{X}$ be an input space and $\mathcal{Y} = \{C_1, ..., C_k\}$ a finite set of classes. We consider a multiclassification task given by a training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ drawn from an unknown distribution $Pr(X,Y)$ from the product $\mathcal{X} \times \mathcal{Y}$. Within this context, we define

Definition 1 *A* nondeterministic hypothesis *is a function h from the input space to the set of non-empty subsets of $\mathcal{Y}$; in symbols, if $\mathscr{P}(\mathcal{Y})$ is the set of all subsets of $\mathcal{Y}$,*

$$h : \mathcal{X} \longrightarrow \mathscr{P}(\mathcal{Y}) \setminus \{\varnothing\}.$$

The aim of such a learning task is to find a nondeterministic hypothesis h from a space $\mathcal{H}$ that optimizes the *expected prediction performance (or risk)* on samples S' independently and identically distributed (i.i.d.) according to the distribution $Pr(X,Y)$

$$R^{\Delta}(h) = \int \Delta(h(x),y)\, d(Pr(x,y)),$$

where $\Delta(h(x),y)$ is a loss function that measures the penalty due to the prediction $h(x)$ when the true value is y.

In nondeterministic classification, we would like to favor those decisions of h that contain the true classes, and a smaller rather than a larger number of classes. In other words, we interpret the output $h(x)$ as an imprecise answer to a query about the right class of an entry $x \in \mathcal{X}$. Thus, nondeterministic classification can be seen as a kind of Information Retrieval task for each entry.

Performance in Information Retrieval is compared using different measures in order to consider different perspectives. The most frequently used measures are *Recall* (proportion of all relevant documents that are found by a search) and *Precision* (proportion of retrieved documents that are relevant). The harmonic average of the two amounts is used to capture the goodness of a hypothesis in a single measure. In the weighted case, the measure is called F_β. The idea is to measure a tradeoff between *Recall* and *Precision*.

For further reference, let us recall the formal definitions of these Information Retrieval measures. Thus, for a prediction of a nondeterministic hypothesis $h(x)$ with $x \in \mathcal{X}$, and a class $y \in \mathcal{Y}$, we can compute the following contingency matrix, where $z \in \mathcal{Y}$,

	$y = z$	$y \neq z$
$z \in h(x)$	a	b
$z \notin h(x)$	c	d

(1)

in which each entry (a,b,c,d) is the number of times that the corresponding combination of memberships occurs. Notice that a can only be 1 or 0, depending on whether the class y is included in the prediction $h(x)$ or not; b is the number of classes different from y included in $h(x)$; $c = 1 - a$; and d is the number of classes different from y that are not included in $h(x)$.

According to the matrix, Equation (1), if h is a nondeterministic hypothesis and $(x,y) \in \mathcal{X} \times \mathcal{Y}$, we thus have the following definitions.

Definition 2 *The* Recall *in a* query *(i.e., an entry x) is defined as the proportion of relevant classes (y) included in $h(x)$:*

$$R(h(x),y) = \frac{a}{a+c} = a = 1_{y \in h(x)}.$$

Definition 3 *The* Precision *is defined as the proportion of retrieved classes in $h(x)$ that are relevant (y):*

$$P(h(x),y) = \frac{a}{a+b} = \frac{1_{y \in h(x)}}{|h(x)|}.$$

$h(x)$	*Precision*	*Recall*	F_1	F_2
$[1,2,3]$	0.33	1	0.50	0.71
$[1,2]$	0.50	1	0.67	0.83
$[1]$	1	1	1	1
$[2,3,4]$	0	0	0	0

Table 1: The *Precision*, *Recall*, F_1, and F_2 for different predictions of a nondeterministic classifier h for an entry x with class 1, $(y = 1)$

In other words, given a hypothesis h, the *Precision* for an entry x, that is, $P(h(x), y)$, is the probability of finding the true class (y) of the entry (x) by randomly choosing one of the classes of $h(x)$.

Finally, the tradeoff is formalized by

Definition 4 *The F_β is defined, in general, by*

$$F_\beta(h(x), y) = \frac{(1+\beta^2)PR}{\beta^2 P + R} = \frac{(1+\beta^2)a}{(1+\beta^2)a + b + \beta^2 c}. \tag{2}$$

Thus, for a nondeterministic classifier h and a pair (x, y),

$$F_\beta(h(x), y) = \begin{cases} \frac{1+\beta^2}{\beta^2+|h(x)|} & \text{if} \quad y \in h(x) \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The most frequently used F-measure is F_1. For ease of reference, let us state that

$$F_1(h(x), y) = \frac{2_{y \in h(x)}}{1 + |h(x)|}.$$

Notice that for deterministic classifiers, the accuracy is equal to *Recall*, *Precision*, and F_β given that $|h(x)| = 1$.

To illustrate the use of the F-measures of an entry, let us consider an example. If we assume that the true class of an entry x is 1, $(y = 1)$, then, depending on the value of $h(x)$, Table 1 reports the *Recall*, *Precision*, F_1, and F_2. We observe that the reward attached to a prediction containing the true class with another extra class ranges from 0.667 for F_1 to 0.833 for F_2; whereas the amounts are lower when the prediction includes 2 extra classes.

Once we have the definition of F_β for individual entries, it is straightforward to extend it to a test set. Hence, when S' is a test set of size n, the average loss on it will be computed by

$$\begin{aligned} R^{\Delta^{ND}}(h, S') &= \frac{1}{n}\sum_{j=1}^{n} \Delta^{ND}(h(x'_j), y'_j) = \frac{1}{n}\sum_{j=1}^{n} \left(1 - F_\beta(h(x'_j), y'_j)\right) \\ &= \frac{1}{n}\sum_{j=1}^{n} \left(1 - \frac{1+\beta^2}{\beta^2 + |h(x'_j)|} 1_{y'_j \in h(x'_j)}\right). \end{aligned} \tag{4}$$

The average *Recall* and *Precision* can be similarly defined. For ease of reference, let us remark that the *Recall* is the proportion of times that $h(x')$ includes y' and is thus a generalization of the *deterministic accuracy*.

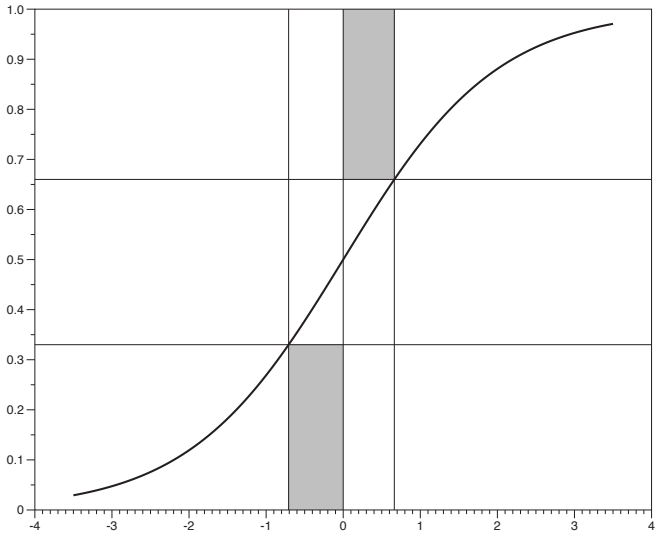


Figure 1: Conditional probabilities of class $+1$ given the discriminant value (horizontal axis) of entries $x \in \mathcal{X}$. Vertical bars separate the region where both classes $\{-1,+1\}$ have a probability of over $1/3$

3.1 Nondeterministic Classification in a Binary Task

To complete this section, let us show what nondeterministic classifiers look like in the simplest case, which will be further developed in the following sections. Let us assume that in a binary classification task (the classes are codified by -1 and $+1$) we have a loss 1 for each false classification. On the other hand, we are allowed to predict both classes, in which case the loss will be $1/3$: the F_1 for a classification of 2 classes containing the true one; see Table 1. The extension for dealing with F_β, with $\beta \neq 1$, is straightforward.

The optimum classifier will return only one class when it is sufficiently sure. In doubtful situations, however, the nondeterministic classifier should opt for predicting the 2 classes. This will be the case whenever the probability of error for both classes is higher than $1/3$, since this is the loss for predictions of two classes; see Figure 1. Therefore, if we have the conditional probabilities of classes given the entries, the optimum classifier will be given by

$$h_{ND}(x) = \begin{cases} \{-1\} & if \quad \eta(x) < 1/3 \\ \{-1,+1\} & if \quad 1/3 \leq \eta(x) < 2/3 \\ \{+1\} & if \quad 2/3 \leq \eta(x), \end{cases} \tag{5}$$

where we are representing by $\eta(x)$ the posterior probability:

$$\eta(x) = Pr(class = +1|x).$$

Notice that Equation (5) is equivalent to the generalized Bayes discriminant function described in Bartlett and Wegkamp (2008) when the cost of using the reject option is calculated using the F_1 loss function.

Algorithm 1 The nondeterministic classifier $nd^{\bullet}$, an algorithm for computing the prediction with one or more classes for an entry x provided that the posterior probabilities of classes are given

Input: $\{C_j : j = 1,..,k \, sorted \, by \, Pr(C_j|x)\}$
Input: β: trade-off between *Recall* and *Precision*
Initialize $i = 0, \Delta_0 = 1$
repeat
  $i = i + 1$
  $\Delta_i = 1 - \frac{1+\beta^2}{\beta^2+i} \sum_{j=1}^{i} Pr(C_j|x)$
until $((i == k)$ or $(\Delta_{i-1} \leq \Delta_i)$
if $(\Delta_{i-1} \leq \Delta_i)$ **then**
  return $\{C_j : j = 1,..,i-1\}$
else
  return $\{C_j : j = 1,..,k\}$
end if

4. Nondeterministic Classification Using Multiclass Posterior Probabilities

In the general multiclass setting presented at the beginning of Section 3, let x be an entry of the input space $\mathcal{X}$ and let us now assume that we know the conditional probabilities of classes given the entry, $Pr(C_j|x)$. Additionally, we shall assume that the classes are ordered according to these probabilities. In this context, we wish to define the

$$h(x) = Z \subset \mathcal{Y} = \{C_1, \ldots, C_k\}$$

that minimizes the risk defined in Equation (1) when we use the nondeterministic loss given by F_β, (Equations 2, 3, and 4). We shall prove that such an $h(x)$ can be computed by Algorithm 1, which does not need to search through all non-empty subsets of $\mathcal{Y}$.

Theorem 1 *(Correctness). If the conditional probabilities $Pr(C_j|x)$ are known, Algorithm 1 returns the nondeterministic prediction for $h(x)$ that minimizes the risk given by the loss $1 - F_\beta$.*

Proof To minimize the risk, Equation (1), it suffices to compute

$$\Delta_x(Z) = \sum_{y \in \mathcal{Y}} \Delta^{ND}(Z, y) Pr(y|x), \tag{6}$$

with $Z \subset \{C_1, \ldots, C_k\}$. Then, we only have to define

$$h(x) = argmin\{\Delta_x(Z) : Z \subset \{C_1, \ldots, C_k\}\}.$$

The proof has two parts. First, we shall see that if $h(x)$ has r classes, then those are the r classes with the highest probabilities; bearing in mind that classes are ordered, $h(x) = Z_r = \{C_j : j = 1,..,r\}$. For this purpose, we need to see that any other subset of r classes will increase the loss due to Z_r. This is a consequence of the following.

The value of Equation (6) for Z_r is Δ_r in Algorithm 1. In fact, with the complementary probability of $\sum_{j=1}^{r} Pr(C_j|x)$, we expect a loss of 1: the *true* class will not be one of the r first classes. On

the other hand, with this sum of probabilities, the *true* class will be in $h(x)$, and therefore the loss will be 1 minus the F_β of the prediction $h(x) = \{C_j : j = 1,..,r\}$:

$$\begin{aligned}\Delta_x(C_j : j = 1,..,r) &= \left(1 - \sum_{j=1}^{r} Pr(C_j|x)\right) + \left(\sum_{j=1}^{r} Pr(C_j|x)\right)\left(1 - \frac{1+\beta^2}{\beta^2+r}\right) \\ &= 1 - \frac{1+\beta^2}{\beta^2+r}\sum_{j=1}^{r} Pr(C_j|x) \\ &= \Delta_r.\end{aligned}$$

Notice that for any other subset of r classes, we could achieve a similar expression simply by modifying the set of posterior probabilities of the last sum. Therefore, to minimize the value of Equation (6) with r classes, we need those with the highest probability.

In the second step, we only have to show that the index r returned by the Algorithm is the right one. We shall see that the search for the best r can be accomplished in linear time, as in the Algorithm. In fact, we shall establish that when the Algorithm reaches the number of classes with which the loss increases, adding further classes will only increase the loss. In symbols, we shall prove that

$$\Delta_r \leq \Delta_{r+1} \Rightarrow \Delta_{r+1} \leq \Delta_{r+2}.$$

To do so, we shall next express the exit condition of the loop $\Delta_r \leq \Delta_{r+1}$ when $(r+1) \leq k$ in a different way. The following expressions are equivalent:

$$\Delta_r \leq \Delta_{r+1} \tag{7}$$

$$\frac{1+\beta^2}{\beta^2+r}\sum_{j=1}^{r} Pr(C_j|x) \geq \frac{1+\beta^2}{\beta^2+r+1}\sum_{j=1}^{r+1} Pr(C_j|x)$$

$$(\beta^2+r+1)\sum_{j=1}^{r} Pr(C_j|x) \geq (\beta^2+r)\sum_{j=1}^{r+1} Pr(C_j|x)$$

$$\sum_{j=1}^{r} Pr(C_j|x) \geq (\beta^2+r)Pr(C_{r+1}|x).$$

Therefore, if $\Delta_r \leq \Delta_{r+1}$ and $(r+1) \leq k$, then

$$Pr(C_{r+1}|x) + \sum_{j=1}^{r} Pr(C_j|x) \geq (\beta^2+r)Pr(C_{r+1}|x) + Pr(C_{r+1}|x).$$

However, bearing in mind that the classes are ordered, we have that $Pr(C_{r+1}|x) \geq Pr(C_{r+2}|x)$, and using Equation (7), we conclude that

$$\sum_{j=1}^{r+1} Pr(C_j|x) \geq (\beta^2+r+1)Pr(C_{r+2}|x) \Leftrightarrow \Delta_{r+1} \leq \Delta_{r+2}.$$

■

4.1 Corollaries

In order to draw some practical consequences, let us reword the previous Theorem. It states that the optimum classification for an input x is the set of r classes with the highest posterior probabilities, where r is the lowest integer that fulfills

$$\sum_{j=1}^{r} Pr(C_j|x) \geq (\beta^2 + r) Pr(C_{r+1}|x), \tag{8}$$

or the set of all classes when this condition is not fulfilled by any r. Expressed in this way, it is straightforward to see that for two classes, with $\beta = 1$, Algorithm 1 coincides with the rule defined in Equation (5).

Additionally, we would like to underscore that Equation (8) hinders the use of naïve *thresholds* to compute nondeterministic predictions. Thus, a nondeterministic classifier that always predicts the top r classes for a constant value r is not a correct option. Equation (8) shows that r, at least, depends on the input x.

Moreover, we should not search for a threshold λ to return, for all inputs, the first r classes whose sum of probabilities is above λ:

$$\sum_{j=1}^{r} Pr(C_j|x) \geq \lambda. \tag{9}$$

Note that given a λ value in $[0,1]$, Equation (9) straightforwardly gives rise to a nondeterministic classifier as follows. For each input x, if the set of classes is ordered according to their posterior probabilities, we define

$$h_\lambda(x) = \left\{ C_1, \ldots, C_r : \sum_{j=1}^{r} Pr(C_j|x) \geq \lambda \quad \& \quad \sum_{j=1}^{r-1} Pr(C_j|x) < \lambda \right\}. \tag{10}$$

Again, the right-hand side of Equation (8) shows that the threshold (λ) would depend on the number of classes predicted, the probability of the first class excluded from the prediction, and the parameter β: the trade-off between *Precision* and *Recall*. The idea behind Equation (8) is that, once we have decided to include the top r classes, to add the $(r+1)^{th}$ class we should guarantee that $Pr(C_{r+1}|x)$ is not much smaller than the sum of probabilities of the top r classes.

However, it may be argued that the inaccuracy of posterior probabilities would partially invalidate the preceding theoretical discussion. In fact, posterior probabilities are not known in practice: they are estimated by algorithms that frequently try to optimize the classification accuracy of a hypothesis that returns the class with the highest probability. In other words, probabilities are discriminant values instead of thorough descriptions of the distribution of classes in a learning task. Therefore, in the experiments reported at the end of the paper, we shall consider the classifiers defined by Equation (10) as a possible alternative method to the nondeterministic classifier of Algorithm 1.

5. Experimental Results

In this section we report the results of a set of experiments conducted to evaluate the proposals of this paper. The next subsection describes the settings used in the experiments: deterministic learners, data sets, procedures to set parameters, and methods to estimate the scores.

Data sets	#classes	#samples	#features
zoo	7	101	16
iris	3	150	4
glass	6	214	9
ecoli	8	336	7
balance scale	3	625	4
vehicle	4	846	18
vowel	11	990	11
contraceptive	3	1473	9
yeast	10	1484	8
car	4	1728	6
image	7	2310	19
waveform	3	5000	40
landsat	6	6435	36
letter recognition	26	20000	16

Table 2: Description of the data sets downloaded from the UCI repository. The classes are not linearly separable

We have two goals here. On the one hand, we compare our approach with two alternative methods. The comparison will first be established with a state-of-the-art set-valued algorithm, the Naïve Credal Classifier *(NCC)* (Zaffalon, 2002; Corani and Zaffalon, 2008a,b). This algorithm is an extension of the traditional Naïve Bayes classifier towards imprecise probabilities and is designed to return robust set-valued (nondeterministic) classifications. We show that our method can improve the performance of *NCC*. We then contrast our method with an implementation of Equation (10); once again our proposals outperform this alternative way to learn nondeterministic classifiers.

On the other hand, we analyze the influence of a number of factors related to nondeterministic learners. We accordingly discuss how the scores of a nondeterministic learner are affected by the quality of posterior probabilities. We see that the performance of a nondeterministic classifier is highly correlated with the accuracy of its deterministic counterpart. The section ends with a study of the meaning of the parameter β.

5.1 Experimental Settings

We used three different methods for learning posterior probabilities in order to build nondeterministic classifiers. First, we employed the Naïve Bayes *(NB)* used by *NCC* as its deterministic counterpart (Corani and Zaffalon, 2008b). The second deterministic learner was a multiclass *SVM*; the implementation used was *libsvm* (Wu et al., 2004) with the linear kernel. Last, we employed the *logistic regression (LR)* of Lin et al. (2008). It should be noted that we are not only using the multiclass classifiers learned by *SVM* or *LR*. Primarily, we apply the mechanisms that provide posterior probabilities from their outputs.

For each of these learners, we built nd^d, where d stands for the name of the deterministic counterpart, *nb*, *svm* or *lr*. Recall that nd^d is the implementation of Algorithm 1 that aims to optimize F_1; that is, $\beta = 1$.

Data sets	#classes	#samples	#features	Original source	Used in
brain	5	42	5597	Pomeroy et al. (2002)	[1]
nci	9	60	7131	Ross et al. (2000)	[1, 3, 4]
lung 6	6	70	16387	Tamayo et al. (2007)	
leukemia 3	3	72	12582	Armstrong et al. (2002)	[2]
lung 4	4	82	9036	Tamayo et al. (2007)	
lung 11	11	89	4459	Tamayo et al. (2007)	
tumors 11	11	174	12533	Su et al. (2001)	[1, 2]
tumors 14	14	190	16063	Ramaswamy et al. (2001)	[1, 2, 4]
lung 16	16	201	493	Tamayo et al. (2007)	
leukemia 7	7	327	12558	Yeoh et al. (2002)	[2]

Table 3: Description of cancer microarray data sets used in the experiments including the original sources and papers from which they are taken. For the sake of brevity, we have denoted the papers as follows: [1] Tibshirani and Hastie (2007), [2] Tan et al. (2005), [3] Staunton et al. (2001), [4] Yeung and Bumgarner (2003)

In the experiments that follow, we used two kinds of data sets. First, we considered data sets downloaded from the UCI repository (Asuncion and Newman, 2007), all of which have more examples than attributes. We included all the data sets that fulfill the following rules: continuous or ordinal attribute values, no more than 40 attributes and no more than 20000 examples. The intention was to consider small data sets that are not linearly separable. Additionally, we excluded those learning tasks with missing values or in which every deterministic learner considered *(NB, SVM, LR)* achieves a proportion of successful classifications of over 95%; otherwise nondeterministic learners would be too similar to their deterministic counterpart. A description of the group of data sets considered can be found in Table 2.

We then evaluated the performance on learning tasks in which the aim was to classify cancer patients from gene expressions captured by microarrays. Unlike the first package of data sets, all the classes are now linearly separable given the dimensions of the input space and the number of entries. Table 3 shows the details of these data sets.

Every table of scores (Tables 4, 5, 6, 7, 8) is devoted to reporting the experimental results achieved in one of the kinds of data sets by one of the deterministic learners and by two nondeterministic algorithms that are to be compared. All the tables have a similar layout. First, they contain the scores of the deterministic learner d: the F_1 (or accuracy or *Recall*), and the Brier score, a measure for the quality of posterior probabilities (Brier, 1950; Yeung et al., 2005), computed by means of

$$BS = \frac{1}{2n} \sum_{i=1}^{n} \sum_{j=1}^{k} \left([y_i = C_j] - Pr(C_j|x_i) \right)^2 .$$

Then we report, for each nondeterministic learner, the F_1, *Precision*, *Recall*, and the average number of classes predicted ($|h(x)|$). All the scores were estimated by means of a 5-fold cross validation repeated 2 times. We did not use the 10-fold procedure, since in certain data sets there are too few examples in some of the classes.

Following Demšar (2006), we used the Wilcoxon signed ranks test to compare the performance of two classifiers when the measurements are F_1, *Precision*, *Recall*, or the average $|h(x)|$. Unless

	NB		*NCC*				nd^{nb}			
Data set	F_1	*BS*	F_1	*P*	*R*	$\lvert h(x)\rvert$	F_1	*P*	*R*	$\lvert h(x)\rvert$
zoo	95.0	0.03	92.3	90.5	100.0	1.496	**95.2**	94.3	97.0	1.055
iris	93.3	0.05	92.9	92.5	94.0	1.037	**93.9**	93.5	94.7	1.023
glass	68.0	0.22	69.2	66.5	76.6	1.321	**70.7**	67.6	77.6	1.253
ecoli	83.5	0.12	82.0	81.0	85.7	1.240	**84.4**	82.2	88.7	1.136
balance	73.9	0.16	76.0	74.2	79.7	1.132	**79.9**	74.9	90.1	1.370
vehicle	60.8	0.30	60.9	59.8	63.4	1.103	**63.3**	60.2	69.6	1.241
vowel	62.1	0.25	64.6	62.6	69.8	1.296	**65.5**	60.9	75.5	1.429
contra	50.0	0.30	50.3	50.1	50.6	1.013	**56.6**	47.9	74.6	1.670
yeast	58.1	0.28	58.4	58.2	59.0	1.037	**60.8**	54.4	74.3	1.500
car	86.8	0.11	**87.3**	87.0	87.8	1.017	83.4	76.6	98.0	1.487
image	90.9	0.08	**91.4**	90.5	94.8	1.195	91.2	90.9	92.0	1.026
waveform	80.1	0.17	80.1	80.0	80.4	1.007	**80.9**	80.0	82.5	1.051
landsat	82.0	0.17	82.0	81.6	83.1	1.058	**82.1**	81.9	82.4	1.011
letter	73.9	0.19	74.6	74.2	75.8	1.081	**74.8**	73.3	78.0	1.166

Table 4: Scores obtained by Naïve Bayes, the Naïve Credal Classifier and nondeterministic classifiers on UCI data sets using a 5-fold cross validation repeated 2 times. For ease of reading, F_1, Precision (*P*), and Recall (*R*) are expressed as percentages. The best nondeterministic F_1 for each data set is boldfaced

explicitly stated, we use the expression statistically *significant differences* to mean that $p < 0.01$. Additionally, in order to provide a quick view of the order of magnitude of the scores, we have boldfaced the best nondeterministic F_1 score for each data set.

To select the regularization parameter, C, for *SVM* and *LR*, we used a 2-fold cross validation repeated 5 times performed on training sets. We searched within $C \in [10^{-2}, \ldots, 10^{2}]$.

5.2 Nondeterministic Classifiers vs. Naïve Credal Classifiers

In this subsection, we compare our nondeterministic learner with *NCC* (Corani and Zaffalon, 2008b), a state-of-the-art set-valued (nondeterministic) algorithm. In order to ensure a fair comparison, our approach uses the Naïve Bayes *(NB)* employed by *NCC* as its deterministic counterpart. Table 4 reports the scores of *NB*, *NCC* and our algorithm nd^{nb}.

The nondeterministic nd^{nb} is significantly (remember that we are using Wilcoxon tests) better than *NCC* both in *Recall* and F_1. Moreover, nd^{nb} wins in 12 out of 14 data sets in F_1, and in 11 out of 14 in *Recall*. However, the scores in *Precision* and size of predictions are more balanced; the differences are not significant. In *Precision*, *NCC* wins in 5 cases, loses in 8, and there is 1 tie situation. The size scores are favorable to *NCC* in 8 out of 14 data sets.

To complete the comparison, we should discuss the results achieved on high dimensional data sets (Table 3). Nevertheless, we do not show the scores on each data set. The characteristics of these tasks are not appropriate for Naïve Bayes (a large number of attributes with a small number of examples); therefore, the posterior probabilities of *NB* are poor (they are significantly worse than those achieved by *SVM* and *LR*) and this affects the performance of our nondeterministic algorithm and *NCC*. Our method tends to be almost deterministic, the average value for the size of predictions is $|h(x)| = 1.008$. This is not optimal, as we shall see later, but it is acceptable behavior. However,

	SVM		nd^{svm}_{λ}				nd^{svm}			
Data set	F_1	*BS*	F_1	P	R	$\lvert h(x)\rvert$	F_1	P	R	$\lvert h(x)\rvert$
zoo	94.0	0.08	38.9	24.6	100.0	4.390	**94.2**	92.4	98.0	1.134
iris	96.0	0.02	83.2	74.8	100.0	1.510	**97.6**	96.7	99.3	1.053
glass	61.7	0.26	**63.7**	53.3	85.0	1.711	63.0	55.9	77.3	1.484
ecoli	86.5	0.11	75.1	66.3	97.2	1.854	**87.4**	85.0	92.3	1.152
balance	91.7	0.06	83.8	77.5	98.7	1.528	**91.3**	89.0	98.1	1.272
vehicle	79.8	0.13	79.6	71.0	97.8	1.576	**82.5**	77.9	92.0	1.297
vowel	82.0	0.15	66.3	55.0	97.5	2.313	**82.9**	78.8	91.5	1.288
contra	51.3	0.29	55.9	48.3	71.3	1.599	**57.7**	46.7	83.1	1.960
yeast	59.0	0.27	60.6	50.4	82.2	1.817	**62.4**	53.4	81.6	1.706
car	85.3	0.11	82.8	76.1	97.3	1.475	**85.6**	83.0	90.8	1.169
image	95.9	0.03	84.8	79.1	99.8	1.579	**96.1**	95.3	97.9	1.058
waveform	86.4	0.10	85.7	80.0	97.1	1.343	**87.6**	81.5	91.8	1.126
landsat	86.8	0.09	84.4	78.4	97.6	1.453	**87.8**	85.7	91.9	1.139
letter	85.8	0.11	71.0	64.3	98.2	2.949	**86.3**	76.7	91.0	1.186

Table 5: Scores obtained by *SVM* learners on UCI data sets using a 5-fold cross validation repeated 2 times. For ease of reading, F_1, Precision (P), and Recall (R) are expressed as percentages. The best nondeterministic F_1 for each data set is boldfaced

	LR		nd^{lr}_{λ}				nd^{lr}			
Data set	F_1	*BS*	F_1	P	R	$\lvert h(x)\rvert$	F_1	P	R	$\lvert h(x)\rvert$
zoo	95.0	0.04	91.0	88.4	97.0	1.252	**95.4**	95.0	96.0	1.045
iris	96.7	0.05	74.4	61.7	100.0	1.767	**94.4**	92.2	99.0	1.137
glass	60.3	0.27	61.5	49.3	86.0	1.844	**63.0**	51.8	85.5	1.774
ecoli	87.5	0.11	76.9	68.3	96.1	1.668	**87.0**	84.4	92.1	1.173
balance	86.7	0.11	**88.9**	87.4	92.6	1.185	88.7	87.7	90.9	1.136
vehicle	77.0	0.16	74.8	64.9	95.3	1.674	**79.2**	74.1	89.7	1.342
vowel	57.9	0.30	54.1	41.5	83.5	2.226	**57.8**	48.6	79.7	1.908
contra	50.8	0.29	55.9	47.7	72.3	1.644	**58.0**	47.4	82.5	1.928
yeast	58.4	0.28	59.4	49.0	80.9	1.818	**61.0**	52.2	79.9	1.713
car	80.9	0.13	80.7	74.1	95.0	1.482	**82.0**	78.9	88.5	1.215
image	88.4	0.11	72.3	60.9	98.7	1.915	**88.0**	85.1	93.8	1.196
waveform	86.5	0.10	81.8	72.8	99.6	1.536	**87.4**	82.9	96.4	1.272
landsat	77.7	0.18	68.6	58.1	93.8	1.940	**76.6**	71.7	86.9	1.387
letter	71.8	0.24	49.3	36.5	90.7	3.253	**70.3**	64.9	82.5	1.556

Table 6: Scores obtained by *LR* learners on UCI data sets using a 5-fold cross validation repeated 2 times. For ease of reading, F_1, Precision (P), and Recall (R) are expressed as percentages. The best nondeterministic F_1 for each data set is boldfaced

the scores of *NCC* on these data sets are inadmissible; their classifiers predict almost all classes for every example, their average values are: $F_1 = 25.73, P = 15.39, R = 100$, and $|h(x)| = 8.58$.

In fact, the behavior of *NCC* is difficult to predict, sometimes it is almost a deterministic classifier, whereas in other tasks the number of classes predicted by *NCC* is very high. Moreover, its degree of nondeterminism is not related to the difficulty of the learning task. When the accuracy of the deterministic classifiers decreases, the average number of classes predicted would be expected

	SVM		nd_{λ}^{svm}				nd^{svm}			
Data set	F_1	*BS*	F_1	*P*	*R*	$\lvert h(x)\rvert$	F_1	*P*	*R*	$\lvert h(x)\rvert$
brain	81.8	0.15	59.2	44.9	97.5	2.504	**82.9**	78.0	93.8	1.401
nci	48.3	0.35	42.9	33.2	68.3	2.492	**47.7**	41.4	65.0	2.167
lung 6	72.1	0.21	65.7	57.9	85.7	1.907	**73.0**	70.4	78.6	1.221
leukemia 3	94.5	0.04	75.1	64.5	100.0	1.862	**95.7**	94.9	97.3	1.049
lung 4	87.1	0.11	73.9	63.0	96.9	1.743	**87.3**	85.3	91.4	1.122
lung 11	58.4	0.31	49.3	36.5	84.2	2.656	**60.4**	53.8	78.0	1.903
tumors 11	89.6	0.13	30.6	19.1	99.7	6.135	**88.9**	87.1	92.8	1.199
tumors 14	70.0	0.26	45.0	35.3	95.0	4.550	**66.5**	60.2	84.7	2.021
lung 16	84.8	0.17	25.0	14.5	100.0	7.440	**87.3**	83.1	95.8	1.266
leukemia 7	92.0	0.07	70.1	59.9	99.4	2.216	**92.1**	90.6	95.1	1.090

Table 7: Scores obtained by *SVM* learners on cancer microarray data sets using a 5-fold cross validation repeated 2 times. For ease of reading, F_1, Precision (*P*), and Recall (*R*) are expressed as percentages. The best nondeterministic F_1 for each data set is boldfaced

	LR		nd_{λ}^{lr}				nd^{lr}			
Data set	F_1	*BS*	F_1	*P*	*R*	$\lvert h(x)\rvert$	F_1	*P*	*R*	$\lvert h(x)\rvert$
brain	86.8	0.11	**86.3**	82.7	94.0	1.274	86.1	84.5	89.2	1.106
nci	55.8	0.33	56.7	55.8	58.3	1.142	**57.8**	56.7	60.0	1.158
lung 6	70.7	0.22	**74.3**	72.5	77.9	1.150	73.3	72.1	75.7	1.107
leukemia 3	97.3	0.02	92.2	88.8	100.0	1.258	**97.7**	97.3	98.7	1.028
lung 4	88.9	0.10	**88.9**	86.4	93.9	1.153	88.8	87.8	90.8	1.061
lung 11	69.0	0.24	**69.1**	65.2	77.5	1.354	68.9	65.7	75.9	1.316
tumors 11	94.8	0.05	89.5	85.9	99.1	1.421	**93.7**	93.0	95.1	1.057
tumors 14	75.3	0.18	**76.8**	73.9	83.2	1.337	**76.8**	75.2	80.3	1.145
lung 16	88.1	0.10	88.3	86.0	93.0	1.157	**88.4**	87.4	90.3	1.060
leukemia 7	91.9	0.07	90.6	87.9	96.3	1.202	**91.9**	91.4	93.1	1.040

Table 8: Scores obtained by *LR* learners on cancer microarray data sets using a 5-fold cross validation repeated 2 times. For ease of reading, F_1, Precision (*P*), and Recall (*R*) are expressed as percentages. The best nondeterministic F_1 for each data set is boldfaced

to increase. However the correlation between the accuracy of *NB* and $|h(x)|$ of *NCC* is 0.24. In the case of nd^{nb}, this correlation is -0.75: negative and quite high.

5.3 Comparing nd with Another Alternative Method

In accordance with the discussion in Section 4.1, we shall now compare the nondeterministic classifiers learned by Algorithm 1 with the alternative classifier defined in Equation (10) that uses a threshold λ for the sum of posterior probabilities. The comparison will be established with posterior probabilities provided by *SVM* and *LR* given that both outperform the accuracy achieved by Naïve Bayes classifiers in the data sets used in these experiments. The λ nondeterministic classifiers will be denoted by nd_{λ}^{d}, where *d* stands for the deterministic counterpart.

To select the parameter λ, we use a grid search employing a 2-fold cross validation repeated 5 times, aiming to optimize F_1. The searching space depends on the learning task *S*. If the proportion of successful classifications for deterministic classifiers, the accuracy, is *a*, then we search within

$\lambda \in [a_0, a_1, \ldots, a_5]$; six options distribute from a to 0.99. In symbols, $a_0 = a, a_5 = 0.99$, and $a_{i+1} - a_i = \frac{0.99-a}{5}$.

In UCI data sets, Tables 5 and 6, nd^{svm} and nd^{lr} win the corresponding nd_λ in 13 out of 14 data sets in F_1 and *Precision*. In *Recall* we have the opposite situation; λ classifiers win in 13 out of 14 cases. Moreover, λ classifiers always predict more classes than nd^{svm} and nd^{lr}. In other words, λ classifiers predict more classes than necessary. All differences are significant. Thus, our *nd* classifiers are better than those computed with the λ parameter.

In cancer microarray data, Tables 7 and 8, nd^{svm} always wins in F_1, *Precision*, and average $|h(x)|$; while nd^{svm} always loses in *Recall*. All differences are again significant. However, when posterior probabilities are provided by *LR*, the differences are not significant in F_1, although nd^{lr} has 5 wins, 1 tie and 4 losses; in *Precision* and average size of predictions the differences are significant in favor of nd^{lr}. Furthermore, as usual, the *Recall* is significantly higher for λ classifiers.

The conclusion is that λ classifiers seem to need more classes in their predictions than *nd* classifiers. In fact, Equation (9) only considers the *Recall*. In practice, this means more *Recall*, but less *Precision* and F_1. Therefore, to optimize the F_1 measure, in an experimental environment, Equation (8) is more adequate than Equation (9), as we have conjectured theoretically in Section 4.1.

5.4 The Importance of Posterior Probabilities

The objective of this subsection is to experimentally investigate the degree of dependency between nondeterministic scores and the accuracy of posterior probabilities. In this study we again employ *SVM* and *LR* with the collection of data sets detailed in Tables 2 and 3.

Let us first consider the set of UCI data sets. Comparing the results in Tables 5 and 6, it can be seen that the scores of nd^{lr} are significantly worse than those of nd^{svm} in F_1, *Precision*, *Recall* ($p < 0.03$), and in average size of predictions. The general message is that nd^{lr} include unnecessary classes in their predictions. The base posterior probabilities seem to be the cause of this behavior: the Brier score of *LR* is significantly worse than that of *SVM*.

On the other hand, the scores obtained with cancer microarray data sets are shown in Tables 7 and 8. The characteristics of UCI and microarray data sets are quite different, and this affects the performance of classifiers. The main difference is that *LR* now has a significantly better Brier score than *SVM*. Moreover, the nd^{lr} algorithm achieves better results than nd^{svm}. The differences are significant in F_1, *Precision*, *Recall* ($p < 0.02$), and average $|h(x)|$. Yet again, inferior posterior probabilities seem to be responsible for the inclusion of unnecessary classes in nondeterministic predictions.

In the preceding discussion of the scores achieved by nondeterministic learners, we found significant differences when the Brier scores of the deterministic counterparts presented significant differences. In fact, the scores of a learner built with Algorithm 1 depend on the quality of the posterior probabilities supplied by the corresponding deterministic learner. It seems plausible to draw the conclusion that the better the posterior probabilities, the better the nondeterministic scores. In order to quantify this statement, we compared deterministic Brier scores with nondeterministic F_1, *Recall*, and *Precision* values; see Figure 2. We separated the scores achieved by UCI and cancer data sets and included the scores of nd^{nb} in UCI data sets. Similar results would be achieved if we compared nondeterministic scores with deterministic accuracy.

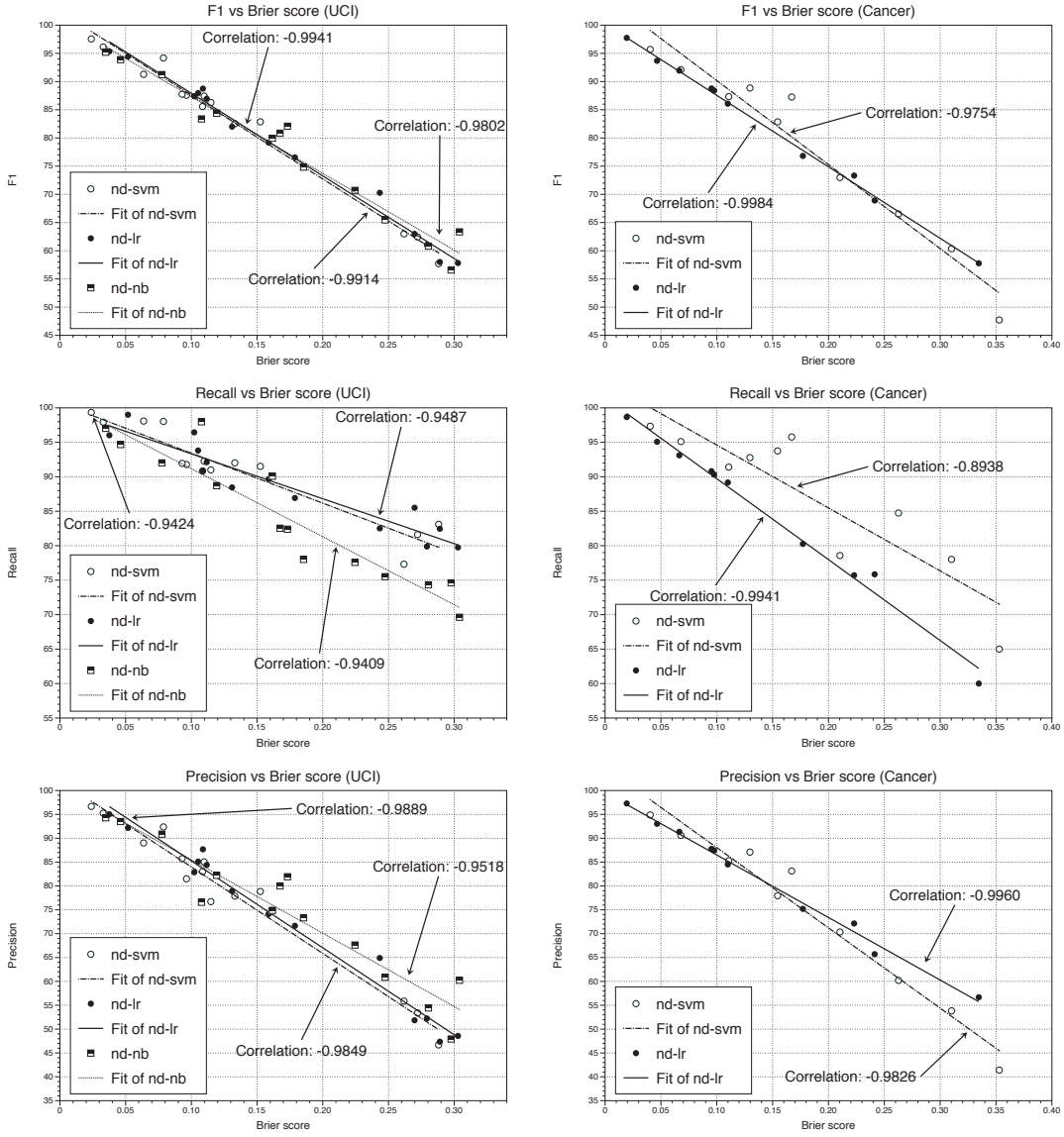


Figure 2: Correlation between Brier scores and F_1, *Recall*, and *Precision*. The left column shows the results with UCI data sets, while the right column uses cancer data sets. Similar results would be achieved if we compared nondeterministic scores with deterministic accuracy

We observed that the correlations between the Brier scores of deterministic learners and nondeterministic scores (F_1, *Recall*, and *Precision*) are very high: their absolute values are in all cases greater than 0.89. Therefore, in order to choose a nondeterministic approach in a practical application, given a data set, it would be recommendable to first analyze the Brier score of different deterministic learners.

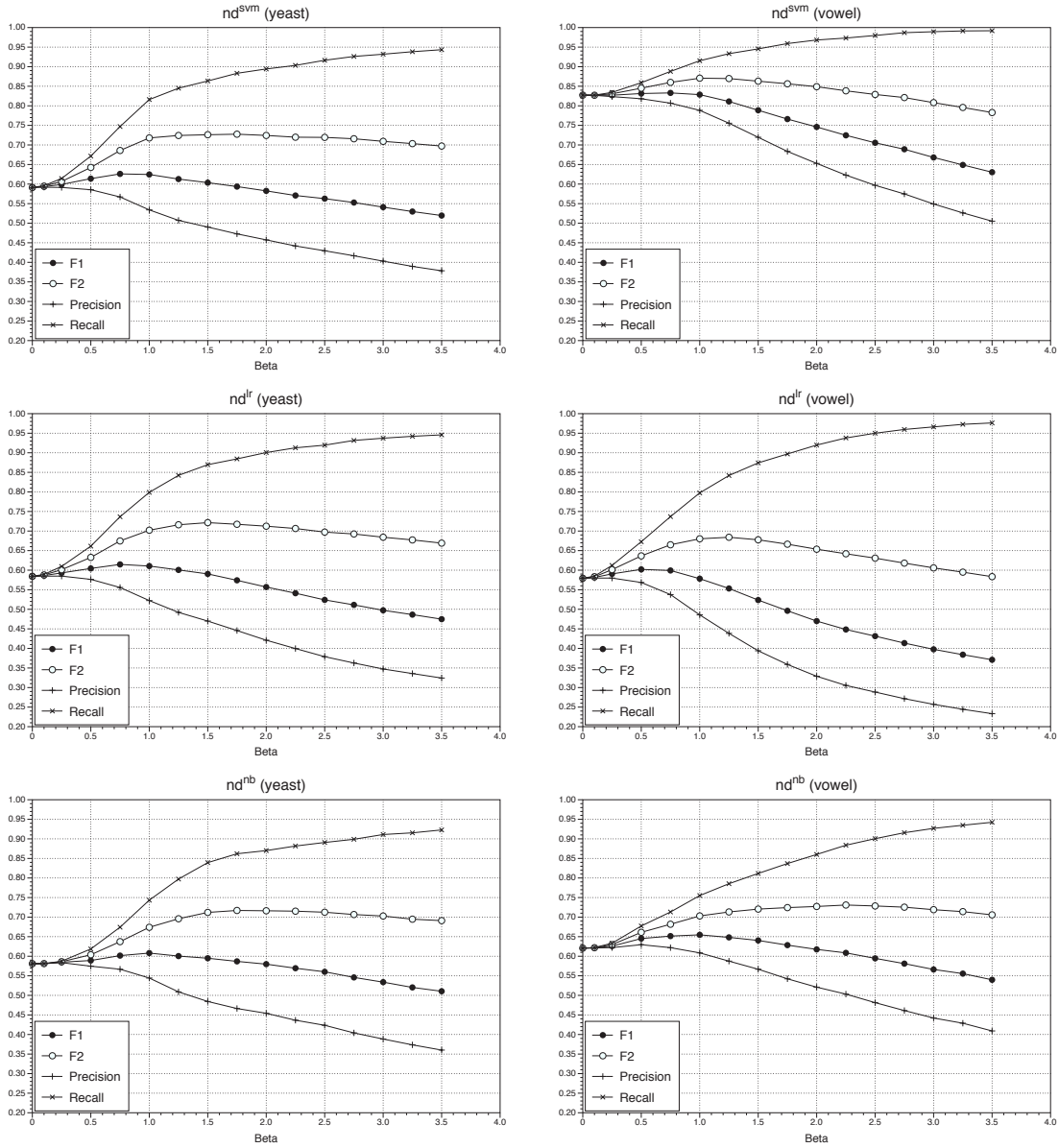


Figure 3: Evolution of F_1, F_2, *Precision* and *Recall* on two UCI data sets (*yeast* and *vowel*) for different β values and for the nondeterministic learners generated by *SVM*, *LR*, and *NB*

5.5 The Meaning of β

In this subsection, we analyze from the point of view of the user the role played by the parameter β in Algorithm 1. Its theoretical aim is to control the size of predictions: as the β value increases, the size of predictions will become bigger and therefore the *Recall* scores will be higher; see Equation (8). The problem is that it is not always of interest to increase *Recall* values, since that would worsen F_1 scores: adding more classes in predictions increases incorrect answers.

In Figure 3 we show the evolution of F_1, F_2, *Precision* and *Recall* on two UCI data sets (*yeast* and *vowel*) for different β values and for the nondeterministic learners generated by *SVM*, *LR*, and

NB. Quite similar graphs could have been generated for the other data sets used in the experiments reported in this section.

Initially, $\beta = 0$ makes the nondeterministic classifiers deterministic. Therefore, the scores represented in the left-hand side of all the graphs in Figure 3 are all the same: the accuracy of the deterministic classifier. As β values become higher, the *Recall* increases and the *Precision* decreases. The main goal of the learning method proposed here is to look for a tradeoff of these measures that is determined by β, a user-modifiable parameter.

In practice, the value of β that the classifier must aim to optimize should be fixed by an expert in the field of application in which the classifier is going to be employed. The kind of decisions that one would like to take from nondeterministic classifications must be considered.

It can be observed in the graphs in Figure 3 that the best scores in F_1 are not always achieved for $\beta = 1$. With small values of β, F_1 increases. However, when some point near 1 is exceeded, the F_1 score of the nondeterministic learner typically falls below the accuracy of the corresponding deterministic learner. Nonetheless, optimal values are frequently reached around the *nominal* value: $\beta = 1$ (or 2 respectively). Slight improvements can be achieved in F_1 (in general F_β) if we use a grid search for β values to be used in Algorithm 1.

6. Conclusions

We have studied classifiers that are allowed to predict more than one class for entries from an input space: nondeterministic or set-valued classifiers. Using a clear analogy with Information Retrieval, we have proposed a family of loss functions based on F_β measures. After discussing such measures, we derived an algorithm to learn optimal nondeterministic hypothesis. Given an entry from the input space, the algorithm requires the posterior probabilities to compute the subset of classes with the lowest expected loss.

The paper includes a set of experiments carried out on two collections of data sets. The first one was downloaded from the UCI repository, the classes of which are not linearly separable. The second group is formed by data sets whose input spaces represent microarray expressions of different kinds of cancer, the classes of which are separable.

Using these benchmarks, we first compared nondeterministic learners obtained from a Naïve Bayes with those learned by a state-of-the-art set-valued (nondeterministic) algorithm, the Naïve Credal Classifier *(NCC)* (Zaffalon, 2002; Corani and Zaffalon, 2008a,b), an extension of the traditional Naïve Bayes classifier designed to return robust set-valued classifications. We showed that, using the loss measures defined in this paper, our method can improve the performance of *NCC*. Additionally, an important advantage of our nondeterministic classifiers over *NCC* is that we can control the degree of nondeterministic behavior. We can regulate the number of classes predicted by fixing the F_β to be optimized: as β is higher (the weight of *Recall* is increased in the harmonic average F_β), the size of our predictions grows (see Section 5.5). However the nondeterministic behavior of *NCC* is quite difficult to predict.

In addition to Naïve Bayes, we used a multiclass *SVM* and a Logistic Regression. With the posterior probabilities provided by these deterministic learners, we built another alternative method to predict more than one class: the set of classes which the highest posterior probabilities summing more than a threshold λ. We also found that the classifiers built with our algorithm outperform this option based on a threshold.

On the other hand, in the experiments reported in this paper, we studied the role of the deterministic learners that explicitly provide posterior probabilities. We found that the better the posterior probabilities, the better the nondeterministic classifiers. In fact we obtained very high correlations between the Brier scores of deterministic probabilities and the F_1, *Precision* and *Recall* values of their nondeterministic counterparts.

Acknowledgments

The research reported here is supported in part under grants TIN2005-08288 from the MEC (Ministerio de Educación y Ciencia, Spain) and TIN2008-06247 from the MICINN (Ministerio de Ciencia e Innovación, Spain). We would also like to acknowledge all those people who generously shared the data sets and software used in this paper, and the anonymous reviewers, whose comments significantly improved it.

References

J. Alonso, J. J. del Coz, J. Díez, O. Luaces, and A. Bahamonde. Learning to predict one or more ranks in ordinal regression tasks. *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD'08)*, LNAI 5211, pages 39–54. Springer, 2008.

S.A. Armstrong, J.E. Staunton, L.B. Silverman, R. Pieters, M.L. den Boer, M.D. Minden, S.E. Sallan, E.S. Lander, T.R. Golub, and S.J. Korsmeyer. MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics*, 30(1):41–47, 2002.

A. Asuncion and D.J. Newman. UCI machine learning repository. *School of Information and Computer Sciences. University of California, Irvine, California, USA*, 2007.

P.L. Bartlett and M.H. Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9:1823–1840, 2008.

G.W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Rev*, 78: 1–3, 1950.

C. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970.

A. Clare and R.D. King. Predicting gene function in Saccharomyces cerevisiae. *Bioinformatics*, 19 (2):42–49, 2003.

G. Corani and M. Zaffalon. Learning reliable classifiers from small or incomplete data sets: The Naive Credal Classifier 2. *Journal of Machine Learning Research*, 9:581–621, 2008a.

G. Corani and M. Zaffalon. JNCC2: The java implementation of Naive Credal Classifier 2. *Journal of Machine Learning Research (Machine Learning Open Source Software)*, 9:2695–2698, 2008b.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

H.P. Kriegel, P. Kroger, A. Pryakhin, and M. Schubert. Using support vector machines for classifying large sets of multi-represented objects. *Proc. 4th SIAM Int. Conf. on Data Mining*, pages 102–114, 2004.

C-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region newton method for logistic regression. *Journal of Machine Learning Research*, 9(Apr):627–650, 2008.

S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. H. Kim, L. C. Goumnerova, P. M. Black, C. Lau, J. C. Allen, D. Zagzag, J. M. Olson, T. Curran, C. Wetmore, J. A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D. N. Louis, J. P. Mesirov, E. S. Lander, and T. R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870):436–442, 2002.

S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J.P. Mesirov, et al. Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences (PNAS)*, 98(26):15149–15154, 2001.

D.T. Ross, U. Scherf, M.B. Eisen, C.M. Perou, C. Rees, P. Spellman, V. Iyer, S.S. Jeffrey, M. Van de Rijn, M. Waltham, et al. Systematic variation in gene expression patterns in human cancer cell lines. *Nature Genetics*, 24(3):227–234, 2000.

G. Shafer and V. Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9:371–421, 2008.

J.E. Staunton, D.K. Slonim, H.A. Coller, P. Tamayo, M.J. Angelo, J. Park, U. Scherf, J.K. Lee, W.O. Reinhold, J.N. Weinstein, et al. Chemosensitivity prediction by transcriptional profiling. *Proceedings of the National Academy of Sciences (PNAS)*, 98(19):10787–10792, 2001.

A.I. Su, J.B. Welsh, L.M. Sapinoso, S.G. Kern, P. Dimitrov, H. Lapp, P.G. Schultz, S.M. Powell, C.A. Moskaluk, H.F. Frierson, and G. M. Hampton. Molecular classification of human carcinomas by use of gene expression signatures. *Cancer Research*, 61(20):7388–7393, 2001.

P. Tamayo, D. Scanfeld, B.L. Ebert, M.A. Gillette, C.W.M. Roberts, and J.P. Mesirov. Metagene projection for cross-platform, cross-species characterization of global transcriptional states. *Proceedings of the National Academy of Sciences (PNAS)*, 104(14):5959–5964, 2007.

A.C. Tan, D.Q. Naiman, L. Xu, R.L. Winslow, and D. Geman. Simple decision rules for classifying human cancers from gene expression profiles. *Bioinformatics*, 21(20):3896–3904, 2005.

R. Tibshirani and T. Hastie. Margin trees for high-dimensional classification. *Journal of Machine Learning Research*, 8:637–652, 2007.

G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, August 2004.

E.J. Yeoh, M.E. Ross, S.A. Shurtleff, W.K. Williams, D. Patel, R. Mahfouz, F.G. Behm, S.C. Raimondi, M.V. Relling, A. Patel, C. Cheng, D. Campana, D. Wilkins, X. Zhou, J. Li, H. Liu, C.-H. Pui, W. E. Evans, C. Naeve, L. Wong, and J. R. Downing. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, 1(2):133–143, 2002.

K.Y. Yeung and R.E. Bumgarner. Multiclass classification of microarray data with repeated measurements: application to cancer. *Genome Biology*, 4(12):R83, 2003.

K.Y. Yeung, R.E. Bumgarner, and A.E. Raftery. Bayesian model averaging: development of an improved multiclass, gene selection and classification tool for microarray data. *Bioinformatics*, 21(10):2394–2402, 2005.

M. Zaffalon. The Naïve Credal Classifier. *Journal of Statistical Planning and Inference*, 105(1): 5–21, 2002.

Journal of Machine Learning Research 10 (2009) 2295-2328 Submitted 3/09; Revised 5/09; Published 10/09

The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs

Han Liu HANLIU@CS.CMU.EDU
John Lafferty LAFFERTY@CS.CMU.EDU
Larry Wasserman LARRY@STAT.CMU.EDU
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA

Editor: Martin J. Wainwright

Abstract

Recent methods for estimating sparse undirected graphs for real-valued data in high dimensional problems rely heavily on the assumption of normality. We show how to use a semiparametric Gaussian copula—or "nonparanormal"—for high dimensional inference. Just as additive models extend linear models by replacing linear functions with a set of one-dimensional smooth functions, the nonparanormal extends the normal by transforming the variables by smooth functions. We derive a method for estimating the nonparanormal, study the method's theoretical properties, and show that it works well in many examples.

Keywords: graphical models, Gaussian copula, high dimensional inference, sparsity, ℓ_1 regularization, graphical lasso, paranormal, occult

1. Introduction

The linear model is a mainstay of statistical inference that has been extended in several important ways. An extension to high dimensions was achieved by adding a sparsity constraint, leading to the lasso (Tibshirani, 1996). An extension to nonparametric models was achieved by replacing linear functions with smooth functions, leading to additive models (Hastie and Tibshirani, 1999). These two ideas were recently combined, leading to an extension called sparse additive models (SpAM) (Ravikumar et al., 2008, 2009a). In this paper we consider a similar nonparametric extension of undirected graphical models based on multivariate Gaussian distributions in the high dimensional setting. Specifically, we use a high dimensional Gaussian copula with nonparametric marginals, which we refer to as a nonparanormal distribution.

If X is a p-dimensional random vector distributed according to a multivariate Gaussian distribution with covariance matrix Σ, the conditional independence relations between the random variables $X_1, X_2, \ldots, X_p$ are encoded in a graph formed from the precision matrix $\Omega = \Sigma^{-1}$. Specifically, missing edges in the graph correspond to zeroes of Ω. To estimate the graph from a sample of size n, it is only necessary to estimate Σ, which is easy if n is much larger than p. However, when p is larger than n, the problem is more challenging. Recent work has focused on the problem of estimating the graph in this high dimensional setting, which becomes feasible if G is sparse. Yuan and Lin (2007)

Assumptions	Dimension	Regression	Graphical Models
parametric	low	linear model	multivariate normal
	high	lasso	graphical lasso
nonparametric	low	additive model	nonparanormal
	high	sparse additive model	ℓ_1-regularized nonparanormal

Figure 1: Comparison of regression and graphical models. The nonparanormal extends additive models to the graphical model setting. Regularizing the inverse covariance leads to an extension to high dimensions, which parallels sparse additive models for regression.

and Banerjee et al. (2008) propose an estimator based on regularized maximum likelihood using an ℓ_1 constraint on the entries of Ω, and Friedman et al. (2007) develop an efficient algorithm for computing the estimator using a graphical version of the lasso. The resulting estimation procedure has excellent theoretical properties, as shown recently by Rothman et al. (2008) and Ravikumar et al. (2009b).

While Gaussian graphical models can be useful, a reliance on exact normality is limiting. Our goal in this paper is to weaken this assumption. Our approach parallels the ideas behind sparse additive models for regression (Ravikumar et al., 2008, 2009a). Specifically, we replace the Gaussian with a semiparametric Gaussian copula. This means that we replace the random variable $X = (X_1, \ldots, X_p)$ by the transformed random variable $f(X) = (f_1(X_1), \ldots, f_p(X_p))$, and assume that $f(X)$ is multivariate Gaussian. This semiparametric copula results in a nonparametric extension of the normal that we call the *nonparanormal* distribution. The nonparanormal depends on the functions $\{f_j\}$, and a mean μ and covariance matrix Σ, all of which are to be estimated from data. While the resulting family of distributions is much richer than the standard parametric normal (the paranormal), the independence relations among the variables are still encoded in the precision matrix $\Omega = \Sigma^{-1}$. We propose a nonparametric estimator for the functions $\{f_j\}$, and show how the graphical lasso can be used to estimate the graph in the high dimensional setting. The relationship between linear regression models, Gaussian graphical models, and their extensions to nonparametric and high dimensional models is summarized in Figure 1.

Most theoretical results on semiparametric copulas focus on low or at least finite dimensional models (Klaassen and Wellner, 1997; Tsukahara, 2005). Models with increasing dimension require a more delicate analysis; in particular, simply plugging in the usual empirical distribution of the marginals does not lead to accurate inference. Instead we use a truncated empirical distribution. We give a theoretical analysis of this estimator, proving consistency results with respect to risk, model selection, and estimation of Ω in the Frobenius norm.

In the following section we review the basic notion of the graph corresponding to a multivariate Gaussian, and formulate different criteria for evaluating estimators of the covariance or inverse covariance. In Section 3 we present the nonparanormal, and in Section 4 we discuss estimation of the model. We present a theoretical analysis of the estimation method in Section 5, with the detailed proofs collected in an appendix. In Section 6 we present experiments with both simulated data and gene microarray data, where the problem is to construct the isoprenoid biosynthetic pathway.

2. Estimating Undirected Graphs

Let $X = (X_1, \ldots, X_p)$ denote a random vector with distribution $P = N(\mu, \Sigma)$. The undirected graph $G = (V, E)$ corresponding to P consists of a vertex set V and an edge set E. The set V has p elements, one for each component of X. The edge set E consists of ordered pairs (i, j) where $(i, j) \in E$ if there is an edge between X_i and X_j. The edge between (i, j) is excluded from E if and only if X_i is independent of X_j given the other variables $X_{\setminus\{i,j\}} \equiv (X_s : \ 1 \leq s \leq p, \ \ s \neq i, j)$, written

$$X_i \perp\!\!\!\perp X_j \ \Big| \ X_{\setminus\{i,j\}}. \tag{1}$$

It is well known that, for multivariate Gaussian distributions, (1) holds if and only if $\Omega_{ij} = 0$ where $\Omega = \Sigma^{-1}$.

Let $X^{(1)}, X^{(2)}, \ldots, X^{(n)}$ be a random sample from P, where $X^{(i)} \in \mathbb{R}^p$. If n is much larger than p, then we can estimate Σ using maximum likelihood, leading to the estimate $\widehat{\Omega} = S^{-1}$, where

$$S = \frac{1}{n} \sum_{i=1}^{n} \left(X^{(i)} - \overline{X} \right) \left(X^{(i)} - \overline{X} \right)^T$$

is the sample covariance, with $\overline{X}$ the sample mean. The zeroes of Ω can then be estimated by applying hypothesis testing to $\widehat{\Omega}$ (Drton and Perlman, 2007, 2008).

When $p > n$, maximum likelihood is no longer useful; in particular, the estimate $\widehat{\Sigma}$ is not positive definite, having rank no greater than n. Inspired by the success of the lasso for linear models, several authors have suggested estimating Σ by minimizing

$$-\ell(\Omega) + \lambda \sum_{j \neq k} |\Omega_{jk}|$$

where

$$\ell(\Omega) = \frac{1}{2} \left(\log |\Omega| - \mathrm{tr}(\Omega S) - p \log(2\pi) \right)$$

is the log-likelihood with S the sample covariance matrix. The estimator $\widehat{\Omega}$ can be computed efficiently using the glasso algorithm (Friedman et al., 2007), which is a block coordinate descent algorithm that uses the standard lasso to estimate a single row and column of Ω in each iteration. Under appropriate sparsity conditions, the resulting estimator $\widehat{\Omega}$ has been shown to have good theoretical properties (Rothman et al., 2008; Ravikumar et al., 2009b).

There are several different ways to judge the quality of an estimator $\widehat{\Sigma}$ of the covariance or $\widehat{\Omega}$ of the inverse covariance. We discuss three in this paper, persistency, norm consistency, and sparsistency. Persistency means consistency in risk, when the model is not necessarily assumed to be correct. Suppose the true distribution P has mean μ_0, and that we use a multivariate normal $p(x; \mu_0, \Sigma)$ for prediction; we do not assume that P is normal. We observe a new vector $X \sim P$ and define the prediction risk to be

$$R(\Sigma) = -\mathbb{E} \log p(X; \mu_0, \Sigma) = -\int \log p(x; \mu_0, \Sigma) \, dP(x).$$

It follows that

$$R(\Sigma) = \frac{1}{2} \left(\mathrm{tr}(\Sigma^{-1} \Sigma_0) + \log |\Sigma| - p \log(2\pi) \right)$$

where Σ_0 is the covariance of X under P. If $\mathcal{S}$ is a set of covariance matrices, the oracle is defined to be the covariance matrix Σ_* that minimizes $R(\Sigma)$ over $\mathcal{S}$:

$$\Sigma_* = \arg\min_{\Sigma \in \mathcal{S}} R(\Sigma).$$

Thus $p(x;\mu_0,\Sigma_*)$ is the best predictor of a new observation among all distributions in $\{p(x;\mu_0,\Sigma) : \Sigma \in \mathcal{S}\}$. In particular, if $\mathcal{S}$ consists of covariance matrices with sparse graphs, then $p(x;\mu_0,\Sigma_*)$ is, in some sense, the best sparse predictor. An estimator $\widehat{\Sigma}_n$ is *persistent* if

$$R(\widehat{\Sigma}_n) - R(\Sigma_*) \xrightarrow{P} 0$$

as the sample size n increases to infinity. Thus, a persistent estimator approximates the best estimator over the class $\mathcal{S}$, but we do not assume that the true distribution has a covariance matrix in $\mathcal{S}$, or even that it is Gaussian. Moreover, we allow the dimension $p = p_n$ to increase with n. On the other hand, norm consistency and sparsistency require that the true distribution is Gaussian. In this case, let Σ_0 denote the true covariance matrix. An estimator is *norm consistent* if

$$\|\widehat{\Sigma}_n - \Sigma\| \xrightarrow{P} 0$$

where $\|\cdot\|$ is a norm. If $E(\Omega)$ denotes the edge set corresponding to Ω, an estimator is *sparsistent* if

$$\mathbb{P}\Big(E(\Omega) \neq E(\widehat{\Omega}_n)\Big) \to 0.$$

Thus, a sparsistent estimator identifies the correct graph consistently. We present our theoretical analysis on these properties of the nonparanormal in Section 5.

3. The Nonparanormal

We say that a random vector $X = (X_1,\ldots,X_p)^T$ has a *nonparanormal* distribution if there exist functions $\{f_j\}_{j=1}^p$ such that $Z \equiv f(X) \sim N(\mu,\Sigma)$, where $f(X) = (f_1(X_1),\ldots,f_p(X_p))$. We then write

$$X \sim NPN(\mu,\Sigma,f).$$

When the f_j's are monotone and differentiable, the joint probability density function of X is given by

$$p_X(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(f(x)-\mu)^T \Sigma^{-1}(f(x)-\mu)\right\} \prod_{j=1}^p |f_j'(x_j)|. \tag{2}$$

Lemma 1 *The nonparanormal distribution $NPN(\mu,\Sigma,f)$ is a Gaussian copula when the f_j's are monotone and differentiable.*

Proof By Sklar's theorem (Sklar, 1959), any joint distribution can be written as

$$F(x_1,\ldots,x_p) = C\{F_1(x_1),\ldots,F_p(x_p)\}$$

where the function C is called a copula. For the nonparanormal we have

$$F(x_1,\ldots,x_p) = \Phi_{\mu,\Sigma}(\Phi^{-1}(F_1(x_1)),\ldots,\Phi^{-1}(F_p(x_p)))$$

where $\Phi_{\mu,\Sigma}$ is the multivariate Gaussian cdf and Φ is the univariate standard Gaussian cdf. Thus, the corresponding copula is

$$C(u_1,\ldots,u_p) = \Phi_{\mu,\Sigma}(\Phi^{-1}(u_1),\ldots,\Phi^{-1}(u_p)).$$

This is exactly a Gaussian copula with parameters μ and Σ. If each f_j is differentiable then the density of X has the same form as (2). ■

Note that the density in (2) is not identifiable; to make the family identifiable we demand that f_j preserve means and variances:

$$\mu_j = \mathbb{E}(Z_j) = \mathbb{E}(X_j) \text{ and } \sigma_j^2 \equiv \Sigma_{jj} = \mathrm{Var}(Z_j) = \mathrm{Var}(X_j). \tag{3}$$

Note that these conditions only depend on $\mathrm{diag}(\Sigma)$ but not the full covariance matrix.

Let $F_j(x)$ denote the marginal distribution function of X_j. Then

$$F_j(x) = \mathbb{P}(X_j \le x) = \mathbb{P}(Z_j \le f_j(x)) = \Phi\left(\frac{f_j(x)-\mu_j}{\sigma_j}\right)$$

which implies that

$$f_j(x) = \mu_j + \sigma_j \Phi^{-1}(F_j(x)). \tag{4}$$

The following basic fact says that the independence graph of the nonparanormal is encoded in $\Omega = \Sigma^{-1}$, as for the parametric normal.

Lemma 2 *If $X \sim NPN(\mu,\Sigma,f)$ is nonparanormal and each f_j is differentiable, then $X_i \perp\!\!\!\perp X_j \,|\, X_{\setminus\{i,j\}}$ if and only if $\Omega_{ij} = 0$, where $\Omega = \Sigma^{-1}$.*

Proof From the form of the density (2), it follows that the density factors with respect to the graph of Ω, and therefore obeys the global Markov property of the graph. ■

Next we show that the above is true for any choice of identification restrictions.

Lemma 3 *Define*

$$h_j(x) = \Phi^{-1}(F_j(x)) \tag{5}$$

and let Λ be the covariance matrix of $h(X)$. Then $X_j \perp\!\!\!\perp X_k \,|\, X_{\setminus\{j,k\}}$ if and only if $\Lambda^{-1}_{jk} = 0$.

Proof We can rewrite the covariance matrix as

$$\Sigma_{jk} = \mathrm{Cov}(Z_j,Z_k) = \sigma_j\sigma_k \mathrm{Cov}(h_j(X_j),h_k(X_k)).$$

Hence $\Sigma = D\Lambda D$ and

$$\Sigma^{-1} = D^{-1}\Lambda^{-1}D^{-1},$$

where D is the diagonal matrix with $\mathrm{diag}(D) = \sigma$. The zero pattern of Λ^{-1} is therefore identical to the zero pattern of Σ^{-1}. ■

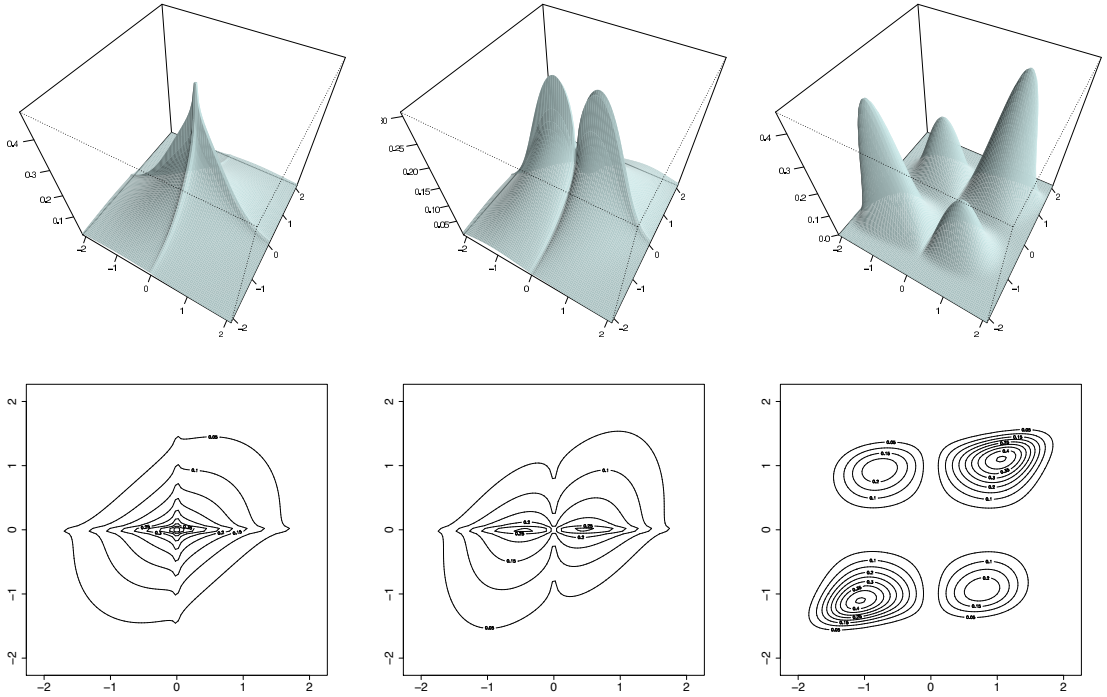


Figure 2: Densities of three 2-dimensional nonparanormals. The component functions have the form $f_j(x) = \text{sign}(x)|x|^{\alpha_j}$. Left: $\alpha_1 = 0.9$, $\alpha_2 = 0.8$; center: $\alpha_1 = 1.2$, $\alpha_2 = 0.8$; right $\alpha_1 = 2, \alpha_2 = 3$. In each case $\mu = (0,0)$ and $\Sigma = \binom{1\ .5}{.5\ 1}$.

Thus, it is not necessary to estimate μ or σ to estimate the graph.

Figure 2 shows three examples of 2-dimensional nonparanormal densities. In each case, the component functions $f_j(x)$ take the form

$$f_j(x) = a_j \text{sign}(x)|x|^{\alpha_j} + b_j$$

where the constants a_j and b_j are set to enforce the identifiability constraints (3). The covariance in each case is $\Sigma = \binom{1\ .5}{.5\ 1}$ and the mean is $\mu = (0,0)$. The exponent α_j determines the nonlinearity. It can be seen how the concavity of the density changes with the exponent α, and that $\alpha > 1$ can result in multiple modes.

The assumption that $f(X) = (f_1(X_1), \ldots, f_p(X_p)$ is normal leads to a semiparametric model where only one dimensional functions need to be estimated. But the monotonicity of the functions f_j, which map onto $\mathbb{R}$, enables computational tractability of the nonparanormal. For more general functions f, the normalizing constant for the density

$$p_X(x) \propto \exp\left\{-\frac{1}{2}(f(x)-\mu)^T \Sigma^{-1}(f(x)-\mu)\right\}$$

cannot be computed in closed form.

4. Estimation Method

Let $X^{(1)},\ldots,X^{(n)}$ be a sample of size n where $X^{(i)} = (X_1^{(i)},\ldots,X_p^{(i)})^T \in \mathbb{R}^p$. In light of (5) we define

$$\widehat{h}_j(x) = \Phi^{-1}(\widetilde{F}_j(x))$$

where $\widetilde{F}_j$ is an estimator of F_j. A natural candidate for $\widetilde{F}_j$ is the marginal empirical distribution function

$$\widehat{F}_j(t) \equiv \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\left\{X_j^{(i)} \le t\right\}}.$$

Now, let θ denote the parameters of the copula. Tsukahara (2005) suggests taking $\widehat{\theta}$ to be the solution of

$$\sum_{i=1}^{n} \phi\left(\widetilde{F}_1(X_1^{(i)}),\ldots,\widetilde{F}_p(X_p^{(i)}),\theta\right) = 0$$

where ϕ is an estimating equation and $\widetilde{F}_j(t) = n\widehat{F}_j(t)/(n+1)$. In our case, θ corresponds to the covariance matrix. The resulting estimator $\widehat{\theta}$, called a rank approximate Z-estimator, has excellent theoretical properties. However, we are interested in the high dimensional scenario where the dimension p is allowed to increase with n; the variance of $\widehat{F}_j(t)$ is too large in this case. Instead, we use the following truncated or *Winsorized*[1] estimator:

$$\widetilde{F}_j(x) = \begin{cases} \delta_n & \text{if } \widehat{F}_j(x) < \delta_n \\ \widehat{F}_j(x) & \text{if } \delta_n \le \widehat{F}_j(x) \le 1-\delta_n \\ (1-\delta_n) & \text{if } \widehat{F}_j(x) > 1-\delta_n, \end{cases} \tag{6}$$

where δ_n is a truncation parameter. Clearly, there is a bias-variance tradeoff in choosing δ_n. Essentially the same estimator with $\delta_n = 1/n$ is studied by Klaassen and Wellner (1997) in the case of bivariate Gaussian copula. In what follows we use

$$\delta_n \equiv \frac{1}{4n^{1/4}\sqrt{\pi \log n}}.$$

This provides the right balance so that we can achieve the desired rate of convergence in our estimate of Ω and the associated undirected graph G in the high dimensional setting.

Given this estimate of the distribution of variable X_j, we then estimate the transformation function f_j by

$$\widetilde{f}_j(x) \equiv \widehat{\mu}_j + \widehat{\sigma}_j \widetilde{h}_j(x) \tag{7}$$

where

$$\widetilde{h}_j(x) = \Phi^{-1}\left(\widetilde{F}_j(x)\right)$$

and $\widehat{\mu}_j$ and $\widehat{\sigma}_j$ are the sample mean and the standard deviation:

$$\widehat{\mu}_j \equiv \frac{1}{n}\sum_{i=1}^{n} X_j^{(i)} \text{ and } \widehat{\sigma}_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_j^{(i)} - \widehat{\mu}_j\right)^2}.$$

1. After Charles P. Winsor, whom John Tukey credited with converting him from topology to statistics Mallows 1990.

Now, let $S_n(\widetilde{f})$ be the sample covariance matrix of $\widetilde{f}(X^{(1)}),\ldots,\widetilde{f}(X^{(n)})$; that is,

$$\begin{aligned} S_n(\widetilde{f}) &\equiv \frac{1}{n}\sum_{i=1}^{n}\left(\widetilde{f}(X^{(i)})-\mu_n(\widetilde{f})\right)\left(\widetilde{f}(X^{(i)})-\mu_n(\widetilde{f})\right)^T \qquad (8)\\ \mu_n(\widetilde{f}) &\equiv \frac{1}{n}\sum_{i=1}^{n}\widetilde{f}(X^{(i)}). \end{aligned}$$

We then estimate Ω using $S_n(\widetilde{f})$. For instance, the maximum likelihood estimator is $\widehat{\Omega}_n^{\mathrm{MLE}} = S_n(\widetilde{f})^{-1}$. The ℓ_1-regularized estimator is

$$\widehat{\Omega}_n = \arg\min_{\Omega}\left\{\mathrm{tr}\left(\Omega S_n(\widetilde{f})\right)-\log|\Omega|+\lambda\|\Omega\|_1\right\} \qquad (9)$$

where λ is a regularization parameter, and $\|\Omega\|_1=\sum_{j\neq k}|\Omega_{jk}|$. The estimated graph is then $\widehat{E}_n = \{(j,k):\ \widehat{\Omega}_{jk}\neq 0\}$.

The nonparanormal is analogous to a sparse additive regression model (Ravikumar et al., 2009a), in the sense that both methods transform the variables by univariate functions. However, while sparse additive models use a regularized risk criterion to fit univariate transformations, our nonparanormal estimator uses a two-step procedure:

1. Replace the observations, for each variable, by their respective normal scores, subject to a Winsorized truncation.

2. Apply the graphical lasso to the transformed data to estimate the undirected graph.

The first step is non-iterative and computationally efficient, with no tuning parameters; it also makes the nonparanormal amenable to theoretical analysis.

Starting with the model in (2), another possibility would be to parametrize each f_j according to some parametric class of monotone functions such as the Box-Cox family, and then find the maximum likelihood estimates of $(\Omega, f_1, \ldots f_p)$ in that class. This might lead to estimates of f_j that depend on Ω, and vice versa, and the estimation problem would not in general be convex. Alternatively, due to (4), the marginal information could be used to estimate the parameters. Our nonparametric approach to estimating the transformations has the advantages of making few assumptions and being easy to compute. In the following section we analyze the theoretical properties of this estimator.

5. Theoretical Results

In this section we present our theoretical results on risk consistency, model selection consistency, and norm consistency of the covariance Σ and inverse covariance Ω. From Lemma 3, the estimate of the graph does not depend on $\sigma_j,\ j\in\{1,\ldots,p\}$ and μ, so we assume that $\sigma_j=1$ and $\mu=0$. Our key technical result is an analysis of the covariance of the Winsorized estimator defined in (6), (7), and (8). In particular, we show that under appropriate conditions,

$$\max_{j,k}\left|S_n(\widetilde{f})_{jk}-S_n(f)_{jk}\right| = o_P(1)$$

where $S_n(\widetilde{f})_{jk}$ denotes the (j,k) entry of the matrix. This result allows us to leverage the recent analysis of Rothman et al. (2008) and Ravikumar et al. (2009b) in the Gaussian case to obtain consistency results for the nonparanormal. More precisely, our main theorem is the following.

Theorem 4 *Suppose that $p = n^{\xi}$ and let $\widetilde{f}$ be the Winsorized estimator defined in* (7) *with* $\delta_n = \frac{1}{4n^{1/4}\sqrt{\pi \log n}}$. *Define*

$$C_M \equiv \frac{48}{\sqrt{\pi}}\left(\sqrt{2M}-1\right)(M+2). \tag{10}$$

For some $M \geq 2(\xi+1)$.

Then for any $\varepsilon \geq C_M \sqrt{\frac{\log p \log^2 n}{n^{1/2}}}$ *and sufficiently large n, we have*

$$\begin{aligned}
&\mathbb{P}\left(\max_{jk}\left|S_n(\widetilde{f})_{jk} - S_n(f)_{jk}\right| > 2\varepsilon\right) \\
&\quad\leq \quad \frac{1}{2\sqrt{\pi \log(np)}} + 2\exp\left(2\log p - \frac{n^{1/2}\varepsilon^2}{1232\pi^2 \log^2 n}\right) + 2\exp\left(2\log p - \frac{n^{1/2}}{8\pi \log n}\right) + o(1).
\end{aligned}$$

The proof of the above theorem is given in Section 7. The following corollary is immediate, and specifies the scaling of the dimension in terms of sample size.

Corollary 5 *Let* $M \geq \max\{15\pi, 2\xi+1\}$. *Then*

$$\mathbb{P}\left(\max_{jk}\left|S_n(\widetilde{f})_{jk} - S_n(f)_{jk}\right| > 2C_M\sqrt{\frac{\log p \log^2 n}{n^{1/2}}}\right) = o(1).$$

Hence,

$$\max_{j,k}\left|S_n(\widetilde{f})_{jk} - S_n(f)_{jk}\right| = O_P\left(\sqrt{\frac{\log p \log^2 n}{n^{1/2}}}\right).$$

The following corollary yields estimation consistency in both the Frobenius norm and the ℓ_2-operator norm. The proof follows the same arguments as the proof of Theorem 1 and Theorem 2 from Rothman et al. (2008), replacing their Lemma 1 with our Theorem 4.

For a matrix $A = (a_{ij})$, the Frobenius norm $\|\cdot\|_F$ is defined as $\|A\|_{\mathrm{F}} \equiv \sqrt{\sum_{i,j} a_{ij}^2}$. The ℓ_2-operator norm $\|\cdot\|_2$ is defined as the magnitude of the largest eigenvalue of the matrix, $\|A\|_2 \equiv \max_{\|x\|_2=1}\|Ax\|_2$. In the following, we write $a_n \asymp b_n$ if there are positive constants c and C independent of n such that $c \leq a_n/b_n \leq C$.

Corollary 6 *Suppose that the data are generated as* $X^{(i)} \sim NPN(\mu_0, \Sigma_0, f_0)$, *and let* $\Omega_0 = \Sigma_0^{-1}$. *If the regularization parameter* λ_n *is chosen as*

$$\lambda_n \asymp 2C_M\sqrt{\frac{\log p \log^2 n}{n^{1/2}}}$$

where C_M *is defined in Theorem 4. Then the nonparanormal estimator* $\widehat{\Omega}_n$ *of* (9) *satisfies*

$$\|\widehat{\Omega}_n - \Omega_0\|_{\mathrm{F}} = O_P\left(\sqrt{\frac{(s+p)(\log p \log^2 n)}{n^{1/2}}}\right)$$

and

$$\|\widehat{\Omega}_n - \Omega_0\|_2 = O_P\left(\sqrt{\frac{s(\log p \log^2 n)}{n^{1/2}}}\right),$$

where

$$s \equiv \mathrm{Card}\left(\{(i,j) \in \{1,\ldots,p\} \times \{1,\ldots,p\} \,|\, \Omega_0(i,j) \neq 0,\ i \neq j\}\right)$$

is the number of nonzero off-diagonal elements of the true precision matrix.

To prove the model selection consistency result, we need further assumptions. We follow Ravikumar (2009) and let the $p^2 \times p^2$ Fisher information matrix of Σ_0 be $\Gamma \equiv \Sigma_0 \otimes \Sigma_0$ where $\otimes$ is the Kronecker matrix product, and define the support set S of $\Omega_0 = \Sigma_0^{-1}$ as

$$S \equiv \{(i,j) \in \{1,\ldots,p\} \times \{1,\ldots,p\} \,|\, \Omega_0(i,j) \neq 0\}.$$

We use S^c to denote the complement of S in the set $\{1,\ldots,p\} \times \{1,\ldots,p\}$, and for any two subsets T and T' of $\{1,\ldots,p\} \times \{1,\ldots,p\}$, we use $\Gamma_{TT'}$ to denote the sub-matrix with rows and columns of Γ indexed by T and T' respectively.

Assumption 1 *There exists some $\alpha \in (0,1]$, such that $\left\|\Gamma_{S^cS}(\Gamma_{SS})^{-1}\right\|_\infty \leq 1-\alpha$.*

As in Ravikumar et al. (2009b), we define two quantities $K_{\Sigma_0} \equiv \|\Sigma_0\|_\infty$ and $K_\Gamma \equiv \|(\Gamma_{SS})^{-1}\|_\infty$. Further, we define the maximum row degree as

$$d \equiv \max_{i=1,\ldots,p} \mathrm{Card}\left(\{j \in 1,\ldots,p \,|\, \Omega_0(i,j) \neq 0\}\right).$$

Assumption 2 *The quantities K_{Σ^0} and K_Γ are bounded, and there are positive constants C such that*

$$\min_{(j,k)\in S} |\Omega_0(j,k)| \geq C\sqrt{\frac{\log^3 n}{n^{1/2}}}$$

for large enough n.

The proof of the following corollary uses our Theorem 4 in place of Equation (12) in the analysis of Ravikumar et al. (2009b).

Corollary 7 *Suppose the regularization parameter is chosen as*

$$\lambda_n \asymp 2C_M\sqrt{\frac{\log p \log^2 n}{n^{1/2}}}$$

where $C(M,n,p)$ is defined in Theorem 4. Then the nonparanormal estimator $\widehat{\Omega}_n$ satisfies

$$\mathbb{P}\left(\mathcal{G}\left(\widehat{\Omega}_n, \Omega_0\right)\right) \geq 1 - o(1)$$

where $\mathcal{G}(\widehat{\Omega}_n, \Omega_0)$ is the event

$$\left\{\mathrm{sign}\left(\widehat{\Omega}_n(j,k)\right) = \mathrm{sign}\left(\Omega_0(j,k)\right), \quad \forall j,k \in S\right\}.$$

Our persistency (risk consistency) result parallels the persistency result for additive models given in Ravikumar et al. (2009a), and allows model dimension that grows exponentially with sample size. The definition in this theorem uses the fact (from Lemma 11) that $\sup_x \Phi^{-1}\left(\widetilde{F}_j(x)\right) \leq \sqrt{2\log n}$ when $\delta_n = 1/(4n^{1/4}\sqrt{\pi \log n})$.

In the next theorem, we do not assume the true model is nonparanormal and define the population and sample risks as

$$\begin{aligned} R(f,\Omega) &= \frac{1}{2}\left\{\mathrm{tr}\left[\Omega \mathbb{E}(f(X)f(X)^T\right] - \log|\Omega| - p\log(2\pi)\right\} \\ \widehat{R}(f,\Omega) &= \frac{1}{2}\left\{\mathrm{tr}\left[\Omega S_n(f)\right] - \log|\Omega| - p\log(2\pi)\right\}. \end{aligned}$$

Theorem 8 *Suppose that $p \leq e^{n^\xi}$ for some $\xi < 1$, and define the classes*

$$\begin{aligned} \mathcal{M}_n &= \left\{f : \mathbb{R} \to \mathbb{R} : f \text{ is monotone with } \|f\|_\infty \leq C\sqrt{\log n}\right\} \\ \mathcal{C}_n &= \left\{\Omega : \|\Omega^{-1}\|_1 \leq L_n\right\}. \end{aligned}$$

Let $\widehat{\Omega}_n$ be given by

$$\widehat{\Omega}_n = \underset{\Omega \in \mathcal{C}_n}{\arg\min}\left\{tr\left(\Omega S_n(\widetilde{f})\right) - \log|\Omega|\right\}.$$

Then

$$R(\widetilde{f}_n, \widehat{\Omega}_n) - \inf_{(f,\Omega)\in \mathcal{M}_n^p \oplus \mathcal{C}_n} R(f,\Omega) = O_P\left(L_n\sqrt{\frac{\log n}{n^{1-\xi}}}\right).$$

Hence the Winsorized estimator of (f,Ω) with $\delta_n = 1/(4n^{1/4}\sqrt{\pi\log n})$ is persistent over $\mathcal{C}_n$ when $L_n = o\left(n^{(1-\xi)/2}/\sqrt{\log n}\right)$.

The proofs of Theorems 4 and 8 are given in Section 7.

6. Experimental Results

In this section, we report experimental results on synthetic and real data sets. We mainly compare the ℓ_1-regularized nonparanormal and Gaussian (paranormal) models, computed using the graphical lasso algorithm (glasso) of Friedman et al. (2007). The primary conclusions are: (i) When the data are multivariate Gaussian, the performance of the two methods is comparable; (ii) when the model is correct, the nonparanormal performs much better than the graphical lasso in many cases; (iii) for a particular gene microarray data set, our method behaves differently from the graphical lasso, and may support different biological conclusions.

Note that we can reuse the glasso implementation to fit a sparse nonparanormal. In particular, after computing the Winsorized sample covariance $S_n(\widetilde{f})$, we pass this matrix to the glasso routine to carry out the optimization

$$\widehat{\Omega}_n = \underset{\Omega}{\arg\min}\left\{\mathrm{tr}\left(\Omega S_n(\widetilde{f})\right) - \log|\Omega| + \lambda_n\|\Omega\|_1\right\}.$$

6.1 Neighborhood Graphs

We begin by describing a procedure to generate graphs as in (Meinshausen and Bühlmann, 2006), with respect to which several distributions can then be defined. We generate a p-dimensional sparse graph $G \equiv (V,E)$ as follows: Let $V = \{1,\ldots,p\}$ correspond to variables $X = (X_1,\ldots,X_p)$. We associate each index j with a point $(Y_j^{(1)}, Y_j^{(2)}) \in [0,1]^2$ where

$$Y_1^{(k)},\ldots,Y_n^{(k)} \sim \text{Uniform}[0,1]$$

for $k = 1,2$. Each pair of nodes (i,j) is included in the edge set E with probability

$$\mathbb{P}\left((i,j) \in E\right) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\|y_i - y_j\|_n^2}{2s}\right)$$

where $y_i \equiv (y_i^{(1)}, y_i^{(2)})$ is the observation of $(Y_i^{(1)}, Y_i^{(2)})$ and $\|\cdot\|_n$ represents the Euclidean distance. Here, $s = 0.125$ is a parameter that controls the sparsity level of the generated graph. We restrict the maximum degree of the graph to be four and build the inverse covariance matrix Ω_0 according to

$$\Omega_0(i,j) = \begin{cases} 1 & \text{if } i = j \\ 0.245 & \text{if } (i,j) \in E \\ 0 & \text{otherwise,} \end{cases}$$

where the value 0.245 guarantees positive definiteness of the inverse covariance matrix.

Given Ω_0, n data points are sampled from

$$X^{(1)},\ldots,X^{(n)} \sim NPN(\mu_0, \Sigma_0, f_0)$$

where $\mu_0 = (1.5,\ldots,1.5)$, $\Sigma_0 = \Omega_0^{-1}$. For simplicity, the transformation functions for all dimensions are the same, $f_1 = \ldots = f_p = f$. To sample data from the nonparanormal distribution, we also require $g \equiv f^{-1}$; two different transformations g are employed.

Definition 9 (Gaussian CDF Transformation) *Let g_0 be a one-dimensional Gaussian cumulative distribution function with mean μ_{g_0} and the standard deviation σ_{g_0}, that is,*

$$g_0(t) \equiv \Phi\left(\frac{t - \mu_{g_0}}{\sigma_{g_0}}\right).$$

We define the transformation function $g_j = f_j^{-1}$ for the j-th dimension as

$$g_j(z_j) \equiv \sigma_j \left(\frac{g_0(z_j) - \int g_0(t)\phi\left(\frac{t-\mu_j}{\sigma_j}\right) dt}{\sqrt{\int \left(g_0(y) - \int g_0(t)\phi\left(\frac{t-\mu_j}{\sigma_j}\right) dt\right)^2 \phi\left(\frac{y-\mu_j}{\sigma_j}\right) dy}} \right) + \mu_j$$

where $\sigma_j = \Sigma_0(j,j)$.

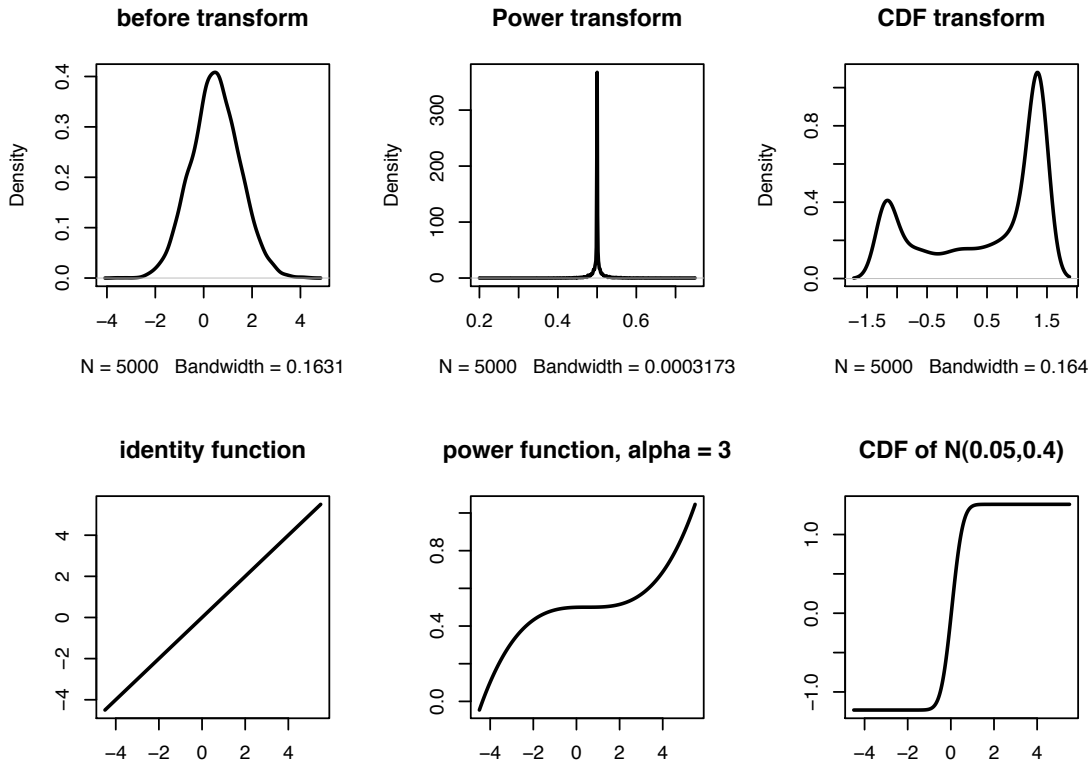


Figure 3: The power and cdf transformations. The densities are estimated using a kernel density estimator with bandwidths selected by cross-validation.

Definition 10 (Symmetric Power Transformation) *Let g_0 be the symmetric and odd transformation given by*

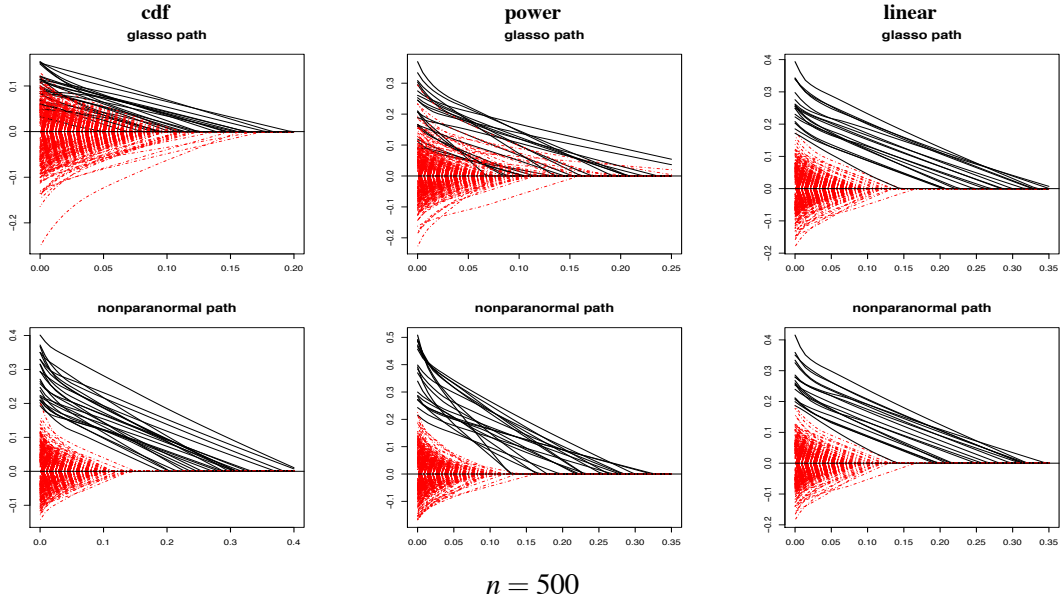
$$g_0(t) = \text{sign}(t)|t|^\alpha$$

where $\alpha > 0$ is a parameter. We define the power transformation for the j-th dimension as

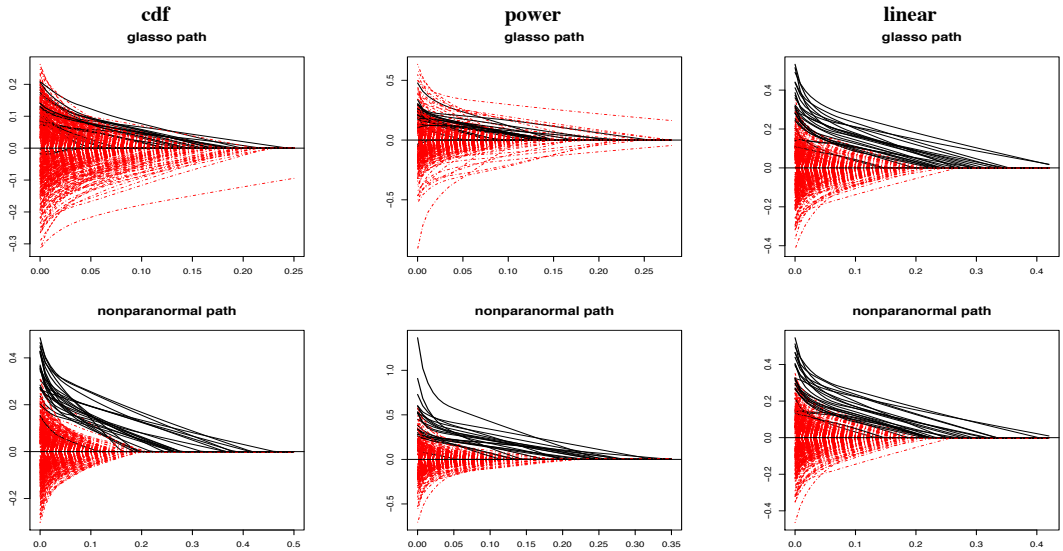
$$g_j(z_j) \equiv \sigma_j \left(\frac{g_0(z_j - \mu_j)}{\sqrt{\int g_0^2(t-\mu_j)\phi\left(\frac{t-\mu_j}{\sigma_j}\right) dt}} \right) + \mu_j.$$

These transformation are constructed to preserve the marginal mean and standard deviation. In the following experiments, we refer to them as the cdf transformation and the power transformation, respectively. For the cdf transformation, we set $\mu_{g_0} = 0.05$ and $\sigma_{g_0} = 0.4$. For the power transformation, we set $\alpha = 3$.

To visualize these two transformations, we sample 5000 data points from a one-dimensional normal distribution $N(0.5, 1.0)$ and then apply the above two transformations; the results are shown in Figure 3. It can be seen how the cdf and power transformations map a univariate normal distribution into a highly skewed and a bi-modal distribution, respectively.



$n = 500$



$n = 200$

Figure 4: Regularization paths for the glasso and nonparanormal with $n = 500$ (top) and $n = 200$ (bottom). The paths for the relevant variables (nonzero inverse covariance entries) are plotted as solid (black) lines; the paths for the irrelevant variables are plotted as dashed (red) lines. For non-Gaussian distributions, the nonparanormal better separates the relevant and irrelevant dimensions.

To generate synthetic data, we set $p = 40$, resulting in $\binom{40}{2} + 40 = 820$ parameters to be estimated, and vary the sample sizes from $n = 200$ to $n = 1000$. Three conditions are considered, corresponding to using the cdf transform, the power transform, or no transformation. In each case, both the glasso and the nonparanormal are applied to estimate the graph.

6.1.1 COMPARISON OF REGULARIZATION PATHS

We choose a set of regularization parameters Λ; for each $\lambda \in \Lambda$, we obtain an estimate $\widehat{\Omega}_n$ which is a 40×40 matrix. The upper triangular matrix has 780 parameters; we vectorize it to get a 780-dimensional parameter vector. A regularization path is the trace of these parameters over all the regularization parameters within Λ. The regularization paths for both methods are plotted in Figure 4. For the cdf transformation and the power transformation, the nonparanormal separates the relevant and the irrelevant dimensions very well. For the glasso, relevant variables are mixed with irrelevant variables. If no transformation is applied, the paths for both methods are almost the same.

6.1.2 ESTIMATED TRANSFORMATIONS

For sample size $n = 1000$, we plot the estimated transformations for three of the variables in Figure 5. It is clear that Winsorization plays a significant role for the power transformation. This is intuitive due to the high skewness of the nonparanormal distribution in this case.

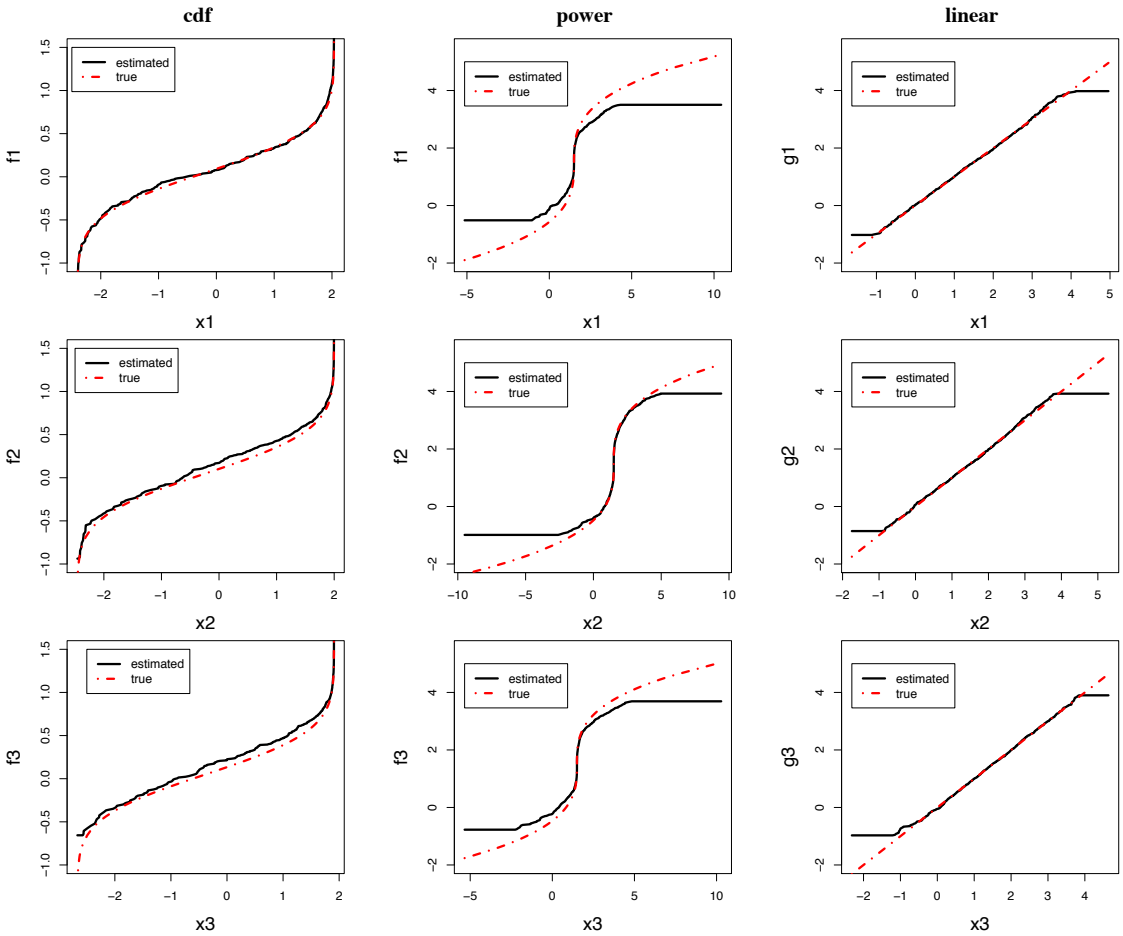


Figure 5: Estimated transformations for the first three variables. Winsorization plays a significant role for the power transformation due to its high skewness.

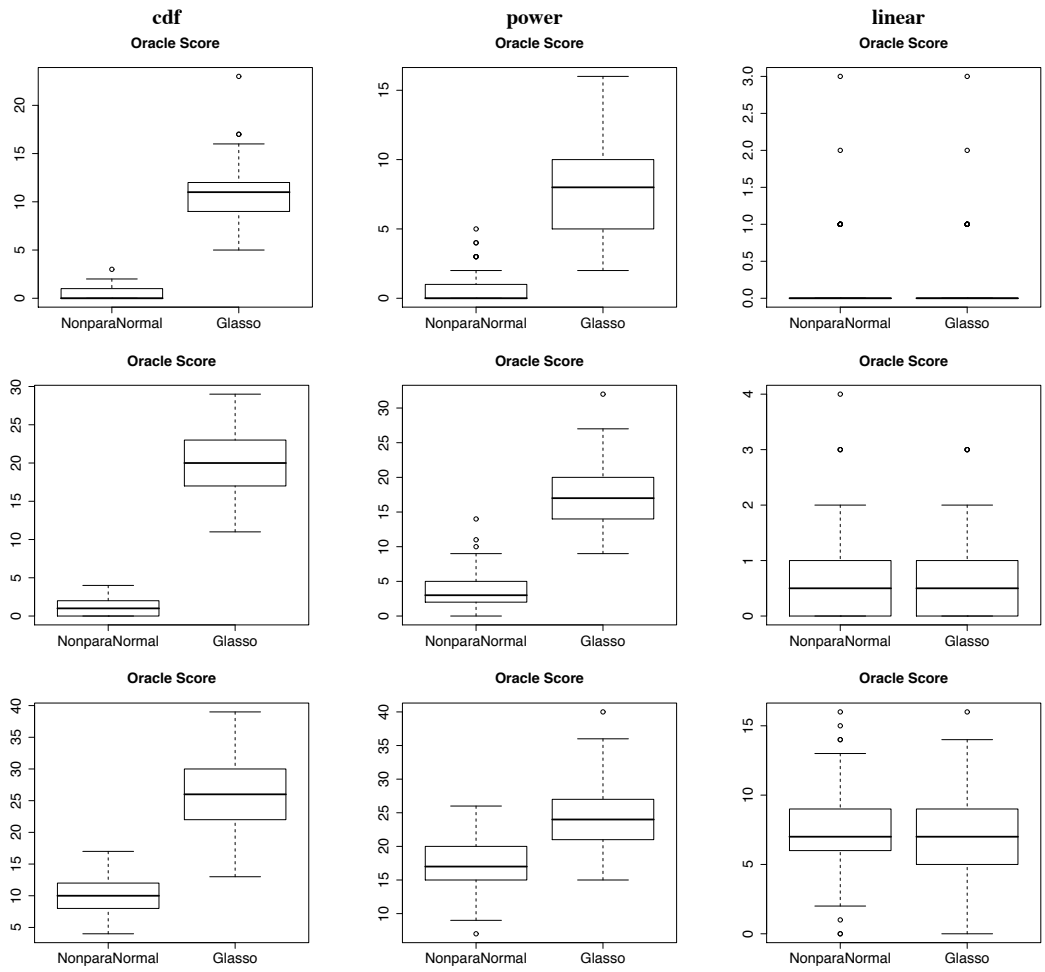


Figure 6: Boxplots of the oracle scores for $n = 1000, 500, 200$ (top, center, bottom).

6.1.3 QUANTITATIVE COMPARISON

To evaluate the performance for structure estimation quantitatively, we use false positive and false negative rates. Let $G = (V, E)$ be a p-dimensional graph (which has at most $\binom{p}{2}$ edges) in which there are $|E| = r$ edges, and let $\widehat{G}^{\lambda} = (V, \widehat{E}^{\lambda})$ be an estimated graph using the regularization parameter λ. The number of false positives at λ is

$$\mathrm{FP}(\lambda) \equiv \text{number of edges in } \widehat{E}^{\lambda} \text{ not in } E$$

The number of false negatives at λ is defined as

$$\mathrm{FN}(\lambda) \equiv \text{number of edges in } E \text{ not in } \widehat{E}^{\lambda}.$$

The oracle regularization level λ^* is then

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \{\mathrm{FP}(\lambda) + \mathrm{FN}(\lambda)\}.$$

The oracle score is $\mathrm{FP}(\lambda^*) + \mathrm{FN}(\lambda^*)$. Figure 6 shows boxplots of the oracle scores for the two methods, calculated using 100 simulations.

To illustrate the overall performance of these two methods over the full paths, ROC curves are shown in Figure 7, using

$$\left(1 - \frac{\mathrm{FN}(\lambda)}{r}, 1 - \frac{\mathrm{FP}(\lambda)}{\binom{p}{2} - r}\right).$$

The curves clearly show how the performance of both methods improves with sample size, and that the nonparanormal is superior to the Gaussian model in most cases.

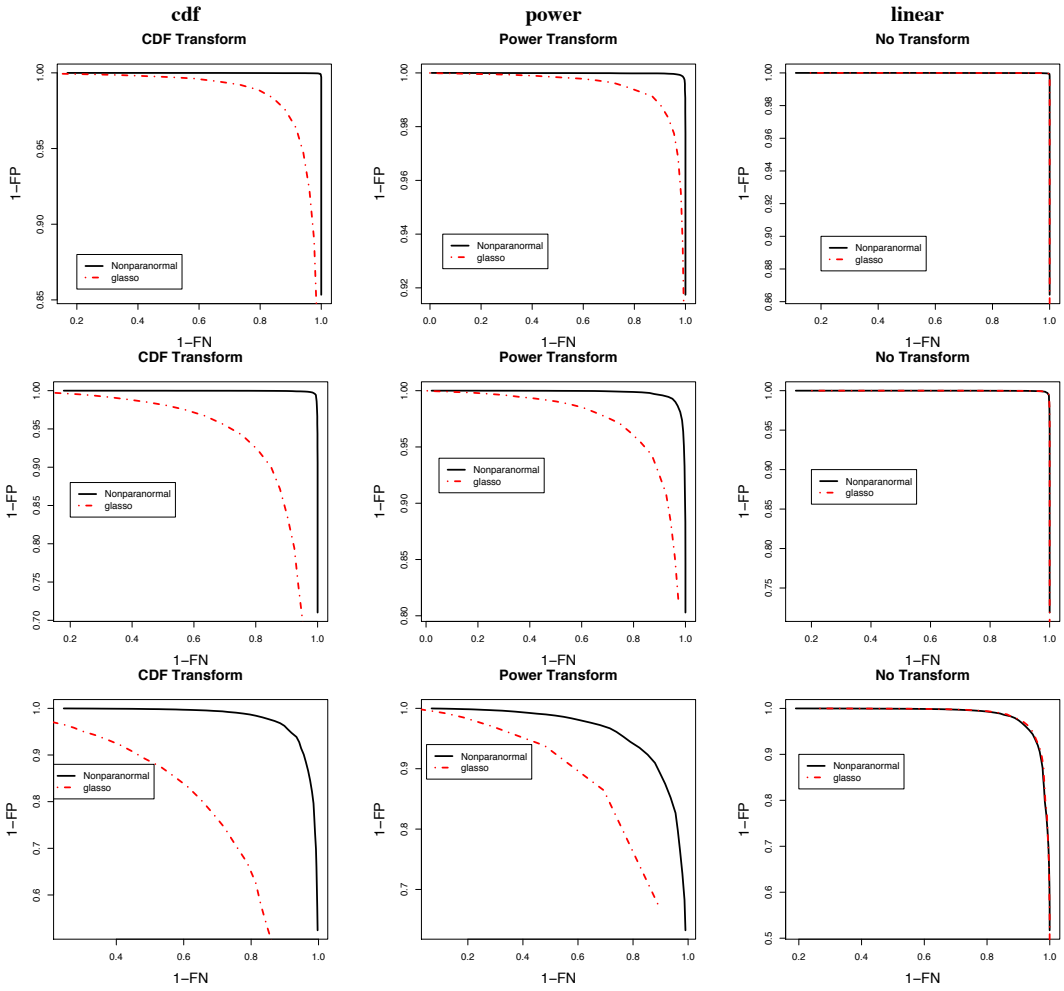


Figure 7: ROC curves for sample sizes $n = 1000, 500, 200$ (top, middle, bottom).

Let $\mathrm{FPE} \equiv \mathrm{FP}(\lambda^*)$ and $\mathrm{FNE} \equiv \mathrm{FN}(\lambda^*)$, Tables 1, 2, and 3 provide numerical comparisons of both methods on data sets with different transformations, where we repeat the experiments 100 times and report the average FPE and FNE values with the corresponding standard deviations. It's clear from the tables that the nonparanormal achieves significantly smaller errors than the glasso if the true distribution of the data is not multivariate Gaussian and achieves performance comparable to the glasso when the true distribution is exactly multivariate Gaussian.

Figure 8 shows typical runs for the cdf and power transformations. It's clear that when the glasso estimates the graph incorrectly, the mistakes include both false positives and negatives.

	Nonparanormal				glasso			
n	FPE	(sd(FPE))	FNE	(sd(FNE))	FPE	(sd(FPE))	FNE	(sd(FNE))
1000	0.10	(0.3333)	0.05	(0.2190)	3.73	(2.3904)	7.24	(3.2910)
900	0.18	(0.5389)	0.16	(0.4197)	3.31	(2.4358)	8.94	(3.2808)
800	0.16	(0.5069)	0.23	(0.5659)	3.80	(2.9439)	9.91	(3.4789)
700	0.26	(0.6295)	0.43	(0.7420)	3.45	(2.5519)	12.26	(3.5862)
600	0.33	(0.6039)	0.41	(0.6371)	3.31	(2.8804)	14.25	(4.0735)
500	0.58	(0.9658)	1.10	(1.0396)	3.18	(2.9211)	17.54	(4.4368)
400	0.71	(1.0569)	1.52	(1.2016)	1.58	(2.3535)	21.18	(4.9855)
300	1.37	(1.4470)	2.97	(2.0123)	0.67	(1.6940)	23.14	(5.0232)
200	2.03	(1.9356)	7.13	(3.4514)	0.01	(0.1000)	24.03	(4.9816)

Table 1: Quantitative comparison on the data set using the cdf transformation. For both FPE and FNE, the nonparanormal performs much better in general.

	Nonparanormal				glasso			
n	FPE	(sd(FPE))	FNE	(sd(FNE))	FPE	(sd(FPE))	FNE	(sd(FNE))
1000	0.27	(0.7086)	0.35	(0.6571)	2.89	(1.9482)	4.97	(2.9213)
900	0.38	(0.6783)	0.41	(0.6210)	2.98	(2.3697)	5.99	(3.0467)
800	0.25	(0.5751)	0.73	(0.8270)	4.10	(2.7834)	6.39	(3.3571)
700	0.69	(0.9067)	0.90	(1.0200)	4.42	(2.8891)	8.80	(3.9848)
600	0.92	(1.2282)	1.59	(1.5314)	4.64	(3.3830)	10.58	(4.2168)
500	1.17	(1.3413)	2.56	(2.3325)	4.00	(2.9644)	13.09	(4.4903)
400	1.88	(1.6470)	4.97	(2.7687)	3.14	(3.4699)	17.87	(4.7750)
300	2.97	(2.4181)	7.85	(3.5572)	1.36	(2.3805)	21.24	(4.7505)
200	2.82	(2.6184)	14.53	(4.3378)	0.37	(0.9914)	24.01	(5.0940)

Table 2: Quantitative comparison on the data set using the power transformation. For both FPE and FNE, the nonparanormal performs much better in general.

6.1.4 COMPARISON IN THE GAUSSIAN CASE

The previous experiments indicate that the nonparanormal works almost as well as the glasso in the Gaussian case. This initially appears surprising, since a parametric method is expected to be more efficient than a nonparametric method if the parametric assumption is correct. To manifest this efficiency loss, we conducted some experiments with very small n and relatively large p. For multivariate Gaussian models, Figure 9 shows results with $(n, p, s) = (50, 40, 1/8), (50, 100, 1/15)$

	Nonparanormal				glasso			
n	FPE	(sd(FPE))	FNE	(sd(FNE))	FPE	(sd(FPE))	FNE	(sd(FNE))
1000	0.10	(0.3333)	0.05	(0.2190)	0.09	(0.3208)	0.06	(0.2386)
900	0.24	(0.7537)	0.14	(0.4025)	0.22	(0.6447)	0.15	(0.4113)
800	0.17	(0.4277)	0.16	(0.3949)	0.16	(0.4431)	0.19	(0.4191)
700	0.25	(0.6871)	0.33	(0.8534)	0.29	(0.8201)	0.27	(0.7501)
600	0.37	(0.7740)	0.36	(0.7456)	0.36	(0.7722)	0.37	(0.6459)
500	0.28	(0.5874)	0.46	(0.7442)	0.25	(0.5573)	0.45	(0.6571)
400	0.55	(0.8453)	1.37	(1.2605)	0.47	(0.7713)	1.35	(1.2502)
300	1.24	(1.3715)	3.07	(1.7306)	0.98	(1.2058)	3.04	(1.8905)
200	1.62	(1.7219)	5.89	(2.7373)	1.55	(1.6779)	5.62	(2.6620)

Table 3: Quantitative comparison on the data set without any transformation. The two methods behave similarly, the glasso is slightly better.

and $(30, 100, 1/15)$. From the mean ROC curves, we see that nonparanormal does indeed behave worse than the glasso, suggesting some efficiency loss. However, from the corresponding boxplots, the efficiency reduction is relatively insignificant.

6.1.5 THE CASE WHEN $p \gg n$

Figure 10 shows results from a simulation of the nonparanormal using cdf transformations with $n = 200, p = 500$ and sparsity level $s = 1/40$. The boxplot shows that the nonparanormal outperforms the glasso. A typical run of the regularization paths confirms this conclusion, showing that the nonparanormal path separates the relevant and irrelevant dimensions very well. In contrast, with the glasso the relevant variables are "buried" among the irrelevant variables.

6.2 Gene Microarray Data

In this study, we consider a data set based on Affymetrix GeneChip microarrays for the plant *Arabidopsis thaliana*, (Wille et al., 2004). The sample size is $n = 118$. The expression levels for each chip are pre-processed by log-transformation and standardization. A subset of 40 genes from the isoprenoid pathway are chosen, and we study the associations among them using both the paranormal and nonparanormal models. Even though these data are generally treated as multivariate Gaussian in the previous analysis (Wille et al., 2004), our study shows that the results of the nonparanormal and the glasso are very different over a wide range of regularization parameters. This suggests the nonparanormal could support different scientific conclusions.

6.2.1 COMPARISON OF THE REGULARIZATION PATHS

We first compare the regularization paths of the two methods, in Figure 11. To generate the paths, we select 50 regularization parameters on an evenly spaced grid in the interval $[0.16, 1.2]$. Although

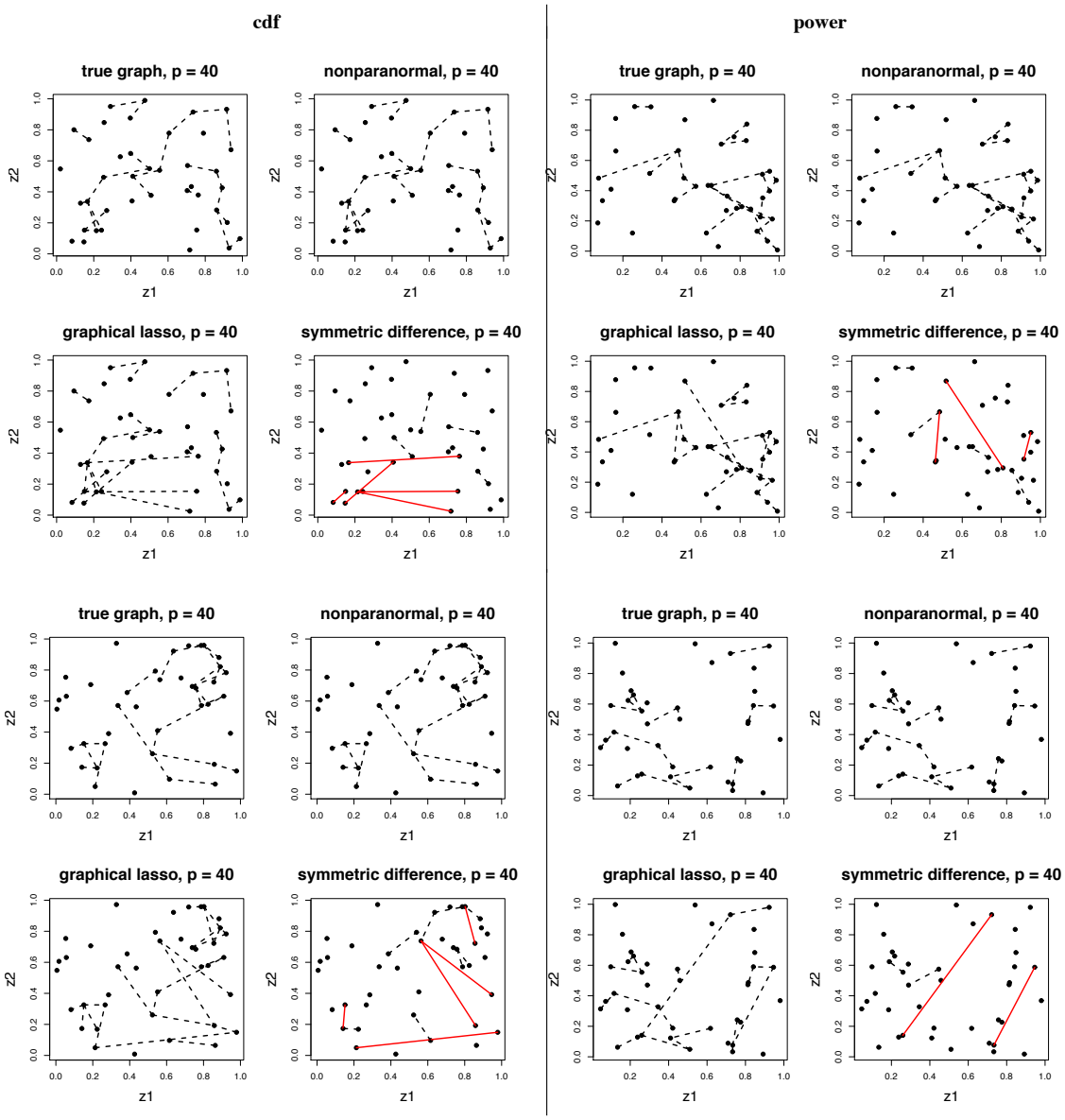


Figure 8: Typical runs for the two methods for $n = 1000$ using the cdf and power transformations. The dashed (black) lines in the symmetric difference plots indicate edges found by the glasso but not the nonparanormal, and vice-versa for the solid (red) lines.

the paths for the two methods look similar, there are some subtle differences. In particular, variables become nonzero in a different order, especially when the regularization parameter is in the range $\lambda \in [0.2, 0.3]$. As shown below, these subtle differences in the paths lead to different model selection behaviors.

6.2.2 COMPARISON OF THE ESTIMATED GRAPHS

Figure 12 compares the estimated graphs for the two methods at several values of the regularization parameter λ in the range $[0.16, 0.37]$. For each λ, we show the estimated graph from the nonparanormal in the first column. In the second column we show the graph obtained by scanning the full

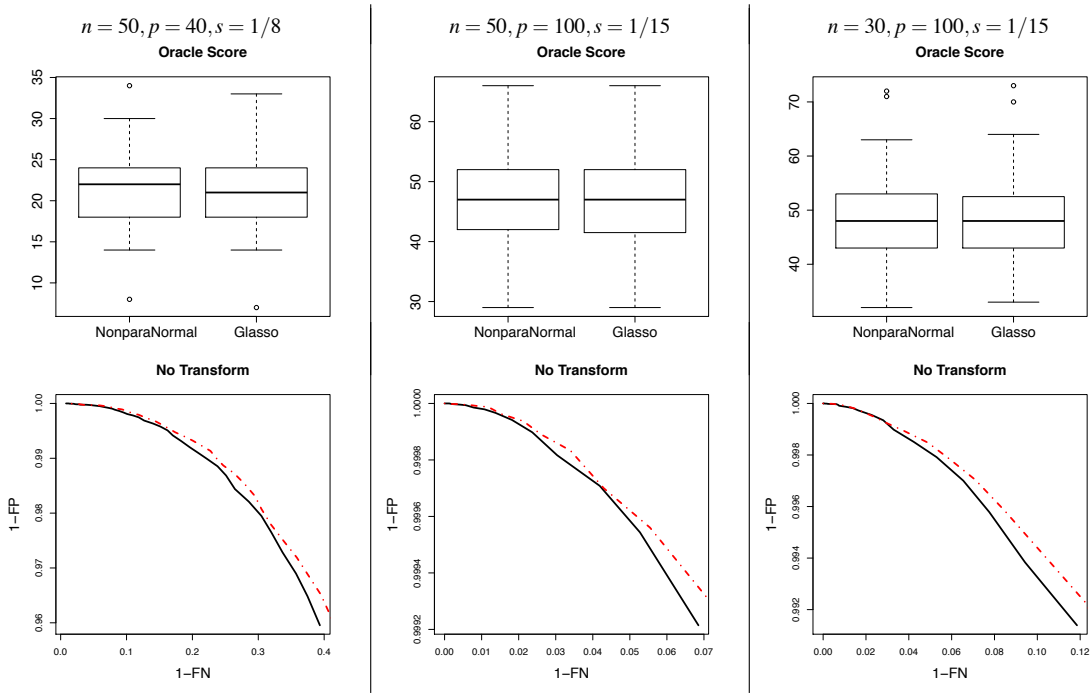


Figure 9: For Gaussian models, comparison of boxplots of the oracle scores and ROC curves for small n and relatively large p. The ROC curves suggest some efficiency loss of the nonparanormal; however, the corresponding boxplots indicate this loss is insignificant.

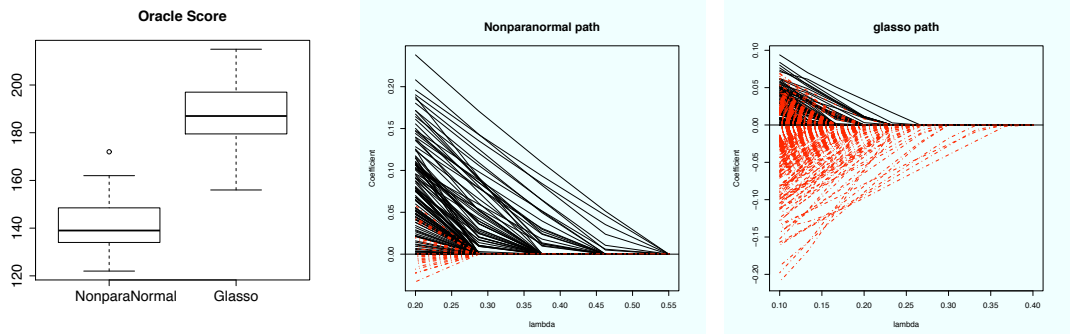


Figure 10: For the cdf transformation with $n = 200, p = 500, s = 1/40$, comparison of the boxplots and a typical run of the regularization paths. The nonparanormal paths separate the relevant from the irrelevant dimensions well. For the glasso, the relevant variables are "buried" in irrelevant variables.

regularization path of the glasso fit and finding the graph having the smallest symmetric difference with the nonparanormal graph. The symmetric difference graph is shown in in the third column. The closest glasso fit is different, with edges selected by the glasso not selected by the nonparanormal, and vice-versa. Several estimated transformations are plotted in Figure 13, which are are nonlinear. Interestingly, several of the differences between the fitted graphs are related to these variables.

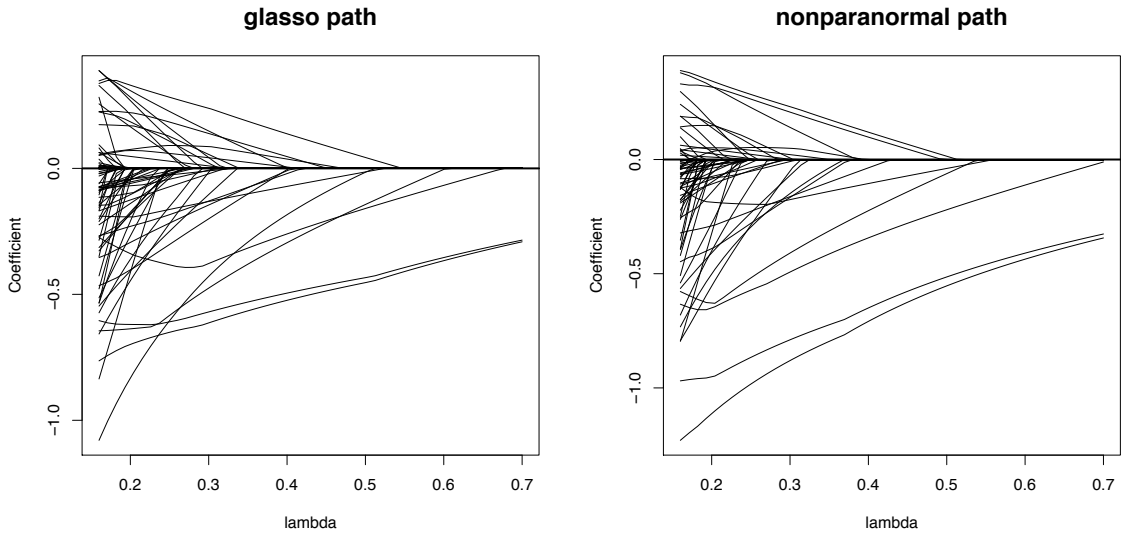


Figure 11: The regularization paths of both methods on the microarray data set. Although the paths for the two methods look similar, there are some subtle differences.

7. Proofs

We assume, without loss of generality from Lemma 3, that $\mu_j = 0$ and $\sigma_j = 1$ for all $j = 1, \ldots, p$. Thus, define $\widetilde{f}_j(x) \equiv \Phi^{-1}(\widetilde{F}_j(x))$ and $f_j(x) \equiv \Phi^{-1}(F_j(x))$, and let $g_j \equiv f_j^{-1}$.

7.1 Proof of Theorem 4

We start with some useful lemmas; the first is from Abramovich et al. (2006).

Lemma 11 *(Gaussian Distribution function vs. Quantile function) Let Φ and ϕ denote the distribution and density functions of a standard Gaussian random variable. Then*

$$\frac{\phi(t)}{2t} \leq 1 - \Phi(t) \leq \frac{\phi(t)}{t} \quad \text{if } t \geq 1$$

and

$$(\Phi^{-1})'(\eta) = \frac{1}{\phi(\Phi^{-1}(\eta))}.$$

Also, for $\eta \geq 0.99$, we have

$$\Phi^{-1}(\eta) = \sqrt{2\log\left(\frac{1}{1-\eta}\right)} - r(\eta) \tag{11}$$

where $r(\eta) \in [0, 1.5]$.

Lemma 12 *(Distribution function of the transformed random variable) For any $\alpha \in (-\infty, \infty)$*

$$\Phi^{-1}\left(F_j\left(g_j(\alpha\sqrt{\log n})\right)\right) = \alpha\sqrt{\log n}.$$

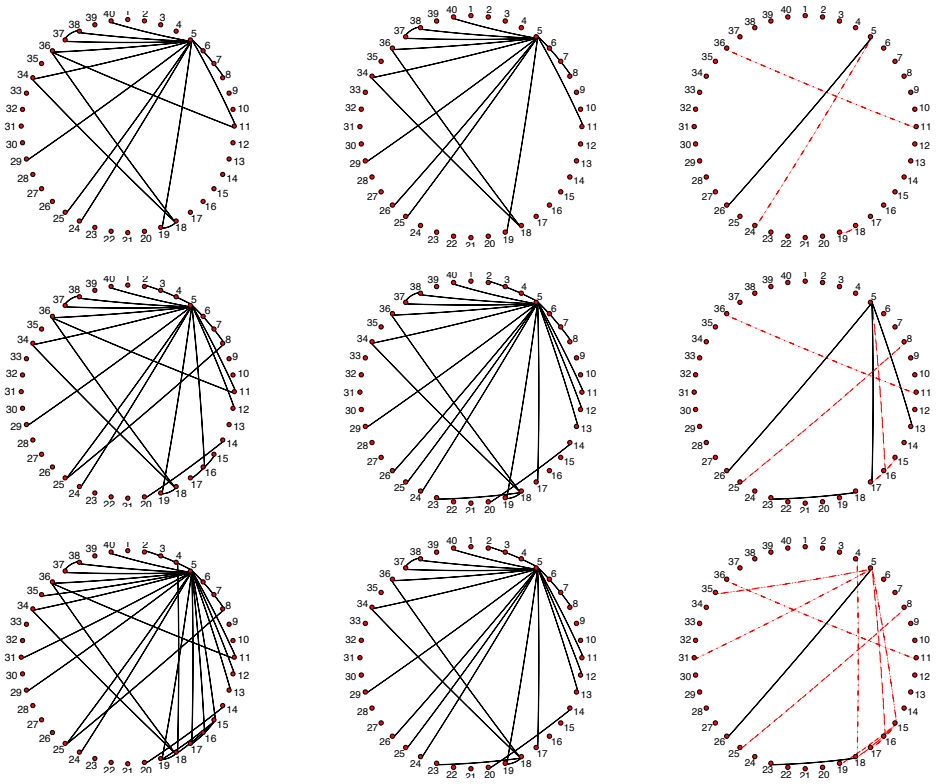


Figure 12: The nonparanormal estimated graph for three values of $\lambda = 0.2448, 0.2661, 0.30857$ (left column), the closest glasso estimated graph from the full path (middle) and the symmetric difference graph (right).

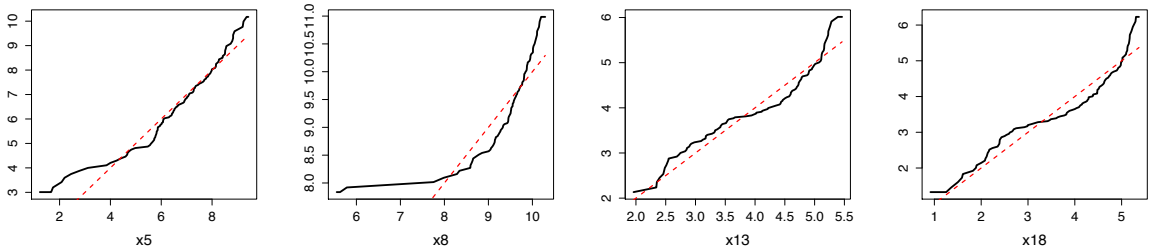


Figure 13: Estimated transformations for the microarray data set, indicating non-Gaussian marginals. The corresponding genes are among the nodes appearing in the symmetric difference graphs above.

Proof The statement follows from

$$F_j(t) = \mathbb{P}(X_j \leq t) = \mathbb{P}(g_j(Z_j) \leq t) = \mathbb{P}(Z_j \leq g_j^{-1}(t)) = \Phi\left(g_j^{-1}(t)\right). \tag{12}$$

which holds for any t. ■

Lemma 13 *(Gaussian maximal inequality) Let $W_1, \ldots, W_n$ be identically distributed standard Gaussian random variables (do not have to be independent). Then for any $\alpha > 0$*

$$\mathbb{P}\left(\max_{1\le i\le n} W_i > \sqrt{\alpha \log n}\right) \le \frac{1}{n^{\alpha/2-1}\sqrt{2\pi\alpha\log n}}.$$

Proof Using Mill's inequality, we have

$$\mathbb{P}\left(\max_{1\le i\le n} W_i > \sqrt{\alpha\log n}\right) \le \sum_{i=1}^{n} \mathbb{P}\left(W_i > \sqrt{\alpha\log n}\right) \le n\frac{\phi(\sqrt{\alpha\log n})}{\sqrt{\alpha\log n}} = \frac{1}{n^{\alpha/2-1}\sqrt{2\pi\alpha\log n}},$$

from which the result follows. ■

Lemma 14 *For any $\alpha > 0$ that satisfies $1-\delta_n - \Phi\left(\sqrt{\alpha\log n}\right) > 0$ for all n, we have*

$$\mathbb{P}\left[\widehat{F}_j\left(g_j\left(\sqrt{\alpha\log n}\right)\right) > 1-\delta_n\right] \le \exp\left\{-2n\left(1-\delta_n-\Phi\left(\sqrt{\alpha\log n}\right)\right)^2\right\}. \tag{13}$$

and

$$\mathbb{P}\left[\widehat{F}_j\left(g_j\left(-\sqrt{\alpha\log n}\right)\right) < \delta_n\right] \le \exp\left\{-2n\left(1-\delta_n-\Phi\left(\sqrt{\alpha\log n}\right)\right)^2\right\}. \tag{14}$$

Proof Using Hoeffding's inequality,

$$\begin{aligned}
&\mathbb{P}\left[\widehat{F}_j\left(g_j\left(\sqrt{\alpha\log n}\right)\right) > 1-\delta_n\right] \\
&= \mathbb{P}\left[\widehat{F}_j\left(g_j\left(\sqrt{\alpha\log n}\right)\right) - F_j\left(g_j\left(\sqrt{\alpha\log n}\right)\right) > 1-\delta_n - F_j\left(g_j\left(\sqrt{\alpha\log n}\right)\right)\right] \\
&\le \exp\left\{-2n\left(1-\delta_n-F_j\left(g_j\left(\sqrt{\alpha\log n}\right)\right)\right)^2\right\}.
\end{aligned}$$

Equation (13) then follows from equation (12). The proof of equation (14) uses the same argument. ■

Now let $M > 2$ and set $\beta = \frac{1}{2}$. We split the interval

$$\left[g_j(-\sqrt{M\log n}), g_j(\sqrt{M\log n})\right]$$

into two parts, the middle

$$\mathcal{M}_n \equiv \left(g_j\left(-\sqrt{\beta\log n}\right), g_j\left(\sqrt{\beta\log n}\right)\right)$$

and ends

$$\mathcal{E}_n \equiv \left[g_j\left(-\sqrt{M\log n}\right), g_j\left(-\sqrt{\beta\log n}\right)\right] \cup \left[g_j\left(\sqrt{\beta\log n}\right), g_j\left(\sqrt{M\log n}\right)\right].$$

The behaviors of the function estimates in these two regions are different, so we first establish bounds on the probability that a sample can fall in the end region $\mathcal{E}_n$.

Lemma 15 *Let* $A \equiv \sqrt{\frac{2}{\pi}}(\sqrt{M} - \sqrt{\beta})$. *Then*

$$\mathbb{P}(X_{1j} \in \mathcal{E}_n) \leq A\sqrt{\frac{\log n}{n^\beta}}, \ \ \forall j \in \{1,\ldots,p\}.$$

Proof Using Equation (12) and the mean value theorem, we have

$$\begin{aligned}
&\mathbb{P}(X_{1j} \in \mathcal{E}_n) \\
&= \mathbb{P}\left(X_{1j} \in \left[g_j(\sqrt{\beta\log n}), g_j(\sqrt{M\log n})\right]\right) + \mathbb{P}\left(X_{1j} \in \left[g_j(-\sqrt{M\log n}), g_j(-\sqrt{\beta\log n})\right]\right) \\
&= F_j\left(g_j(\sqrt{M\log n})\right) - F_j\left(g_j(\sqrt{\beta\log n})\right) + F_j\left(g_j(-\sqrt{\beta\log n})\right) - F_j\left(g_j(-\sqrt{M\log n})\right) \\
&= 2\left(\Phi(\sqrt{M\log n}) - \Phi(\sqrt{\beta\log n})\right) \\
&\leq 2\phi\left(\sqrt{\beta\log n}\right)\left(\sqrt{M\log n} - \sqrt{\beta\log n}\right).
\end{aligned}$$

The result of the lemma follows directly. ■

We next bound the error of the Winsorized estimate of a component function over the end region.

Lemma 16 *For all n, we have*

$$\sup_{t\in\mathcal{E}_n}\left|\Phi^{-1}(\widetilde{F}_j(t)) - \Phi^{-1}(F_j(t))\right| < \sqrt{2(M+2)\log n}, \ \ \forall j \in \{1,\ldots,p\}.$$

Proof From Lemma 12 and the definition of $\mathcal{E}_n$, we have

$$\sup_{t\in\mathcal{E}_n}\left|\Phi^{-1}(F_j(t))\right| \in \left[0, \sqrt{M\log n}\right].$$

Given the fact that $\delta_n = \dfrac{1}{4n^{1/4}\sqrt{\pi\log n}}$, we have $\widetilde{F}_j(t) \in \left(\dfrac{1}{n}, 1-\dfrac{1}{n}\right)$. Therefore, from Equation (11),

$$\sup_{t\in\mathcal{E}_n}\left|\Phi^{-1}\left(\widetilde{F}_j(t)\right)\right| \in \left[0, \sqrt{2\log n}\right).$$

The result follows from the triangle inequality and $\sqrt{M} + \sqrt{2} \leq \sqrt{2(M+2)}$. ■

Now for any $\varepsilon > 0$, we have

$$\begin{aligned}
&\mathbb{P}\left(\max_{j,k}\left|S_n(\widetilde{f})_{jk} - S_n(f)_{jk}\right| > 2\varepsilon\right) \\
&= \mathbb{P}\left(\max_{j,k}\left|\frac{1}{n}\sum_{i=1}^n\left\{\widetilde{f}_j(X_{ij})\widetilde{f}_k(X_{ik}) - f_j(X_{ij})f_k(X_{ik}) - \mu_n(\widetilde{f}_j)\mu_n(\widetilde{f}_k) + \mu_n(f_j)\mu_n(f_k)\right\}\right| > 2\varepsilon\right) \\
&\leq \mathbb{P}\left(\max_{j,k}\left|\frac{1}{n}\sum_{i=1}^n\left(\widetilde{f}_j(X_{ij})\widetilde{f}_k(X_{ik}) - f_j(X_{ij})f_k(X_{ik})\right)\right| > \varepsilon\right) \\
&\quad + \mathbb{P}\left(\max_{j,k}\left|\mu_n(\widetilde{f}_j)\mu_n(\widetilde{f}_k) - \mu_n(f_j)\mu_n(f_k)\right| > \varepsilon\right).
\end{aligned}$$

We only need to analyze the rate for the first term above, since the second one is of higher order (Cai et al., 2008). Let

$$\Delta_i(j,k) \equiv \widetilde{f}_j(X_{ij})\widetilde{f}_k(X_{ik}) - f_j(X_{ij})f_k(X_{ik})$$

and

$$\Theta_{t,s}(j,k) \equiv \widetilde{f}_j(t)\widetilde{f}_k(s) - f_j(t)f_k(s).$$

We define the event $\mathcal{A}_n$ as

$$\mathcal{A}_n \equiv \left\{ g_j\left(-\sqrt{M\log n}\right) \le X_{1j},\ldots,X_{nj} \le g_j\left(\sqrt{M\log n}\right), j=1,\ldots,p \right\}.$$

Then, by Lemma 13, when $M \ge 2(\xi+1)$, we have

$$\mathbb{P}(\mathcal{A}_n^c) \le \mathbb{P}\left(\max_{i,j\in\{1,\ldots,n\}\times\{1,\ldots,p\}} |f_j(X_{ij})| > \sqrt{2\log(np)} \right) \le \frac{1}{2\sqrt{\pi\log(np)}}.$$

Therefore

$$\begin{aligned}
\mathbb{P}\left(\max_{j,k} \left| \frac{1}{n}\sum_{i=1}^n \Delta_i(j,k) \right| > \varepsilon \right) &\le \mathbb{P}\left(\max_{j,k} \left| \frac{1}{n}\sum_{i=1}^n \Delta_i(j,k) \right| > \varepsilon, \mathcal{A}_n \right) \quad + \quad \mathbb{P}(\mathcal{A}_n^c) \\
&\le \mathbb{P}\left(\max_{j,k} \left| \frac{1}{n}\sum_{i=1}^n \Delta_i(j,k) \right| > \varepsilon, \mathcal{A}_n \right) \quad + \quad \frac{1}{2\sqrt{\pi\log(np)}}.
\end{aligned}$$

Thus, we only need to carry out our analysis on the event $\mathcal{A}_n$. On this event, we have the following decomposition:

$$\begin{aligned}
&\mathbb{P}\left(\max_{j,k} \left| \frac{1}{n}\sum_{i=1}^n \Delta_i(j,k) \right| > \varepsilon,\ \mathcal{A}_n \right) \\
&\le \quad \mathbb{P}\left(\max_{j,k} \frac{1}{n} \sum_{X_{ij}\in\mathcal{M}_n, X_{ik}\in\mathcal{M}_n} |\Delta_i(j,k)| > \frac{\varepsilon}{4} \right) + \mathbb{P}\left(\max_{j,k} \frac{1}{n} \sum_{X_{ij}\in\mathcal{L}_n, X_{ik}\in\mathcal{L}_n} |\Delta_i(j,k)| > \frac{\varepsilon}{4} \right) \\
&\quad + 2\mathbb{P}\left(\max_{j,k} \frac{1}{n} \sum_{X_{ij}\in\mathcal{M}_n, X_{ik}\in\mathcal{L}_n} |\Delta_i(j,k)| > \frac{\varepsilon}{4} \right).
\end{aligned}$$

We now analyze each of these terms separately.

Lemma 17 *On the event $\mathcal{A}_n$, let $\beta = 1/2$ and $\varepsilon \ge C_M \sqrt{\dfrac{\log p \log^2 n}{n^{1/2}}}$, then*

$$\mathbb{P}\left(\max_{j,k} \frac{1}{n} \sum_{X_{ij}\in\mathcal{L}_n, X_{ik}\in\mathcal{L}_n} |\Delta_i(j,k)| > \frac{\varepsilon}{4} \right) = o(1).$$

Proof We define

$$\theta_1 \equiv \frac{n^{\beta/2}\varepsilon}{8A\sqrt{\log n}}$$

with the same parameter A as in Lemma 15. Such a θ_1 guarantees that

$$\frac{n\varepsilon}{4\theta_1} - nA\sqrt{\frac{\log n}{n^\beta}} = nA\sqrt{\frac{\log n}{n^\beta}} > 0.$$

By Lemma 15, we have

$$\begin{aligned}
\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^n \mathbf{1}_{\{X_{ij}\in\mathcal{E}_n, X_{ik}\in\mathcal{E}_n\}} > \frac{\varepsilon}{4\theta_1}\right) &\le \mathbb{P}\left(\sum_{i=1}^n \mathbf{1}_{\{X_{ij}\in\mathcal{E}_n\}} > \frac{n\varepsilon}{4\theta_1}\right) \\
&= \mathbb{P}\left(\sum_{i=1}^n \left(\mathbf{1}_{\{X_{ij}\in\mathcal{E}_n\}} - \mathbb{P}(X_{1j}\in\mathcal{E}_n)\right) > \frac{n\varepsilon}{4\theta_1} - n\mathbb{P}(X_{1j}\in\mathcal{E}_n)\right) \\
&\le \mathbb{P}\left(\sum_{i=1}^n \left(\mathbf{1}_{\{X_{ij}\in\mathcal{E}_n\}} - \mathbb{P}(X_{1j}\in\mathcal{E}_n)\right) > \frac{n\varepsilon}{4\theta_1} - nA\sqrt{\frac{\log n}{n^\beta}}\right).
\end{aligned}$$

Using the Bernstein's inequality, for $\beta = \frac{1}{2}$,

$$\begin{aligned}
\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^n \mathbf{1}_{\{X_{ij}\in\mathcal{E}_n, X_{ik}\in\mathcal{E}_n\}} > \frac{\varepsilon}{4\theta_1}\right) &\le \mathbb{P}\left(\sum_{i=1}^n \left(\mathbf{1}_{\{X_{ij}\in\mathcal{E}_n\}} - \mathbb{P}(X_{1j}\in\mathcal{E}_n)\right) > nA\sqrt{\frac{\log n}{n^\beta}}\right) \\
&\le \exp\left(-\frac{c_1 n^{2-\beta}\log n}{c_2 n^{1-\beta/2}\sqrt{\log n} + c_3 n^{1-\beta/2}\sqrt{\log n}}\right) = o(1),
\end{aligned}$$

where $c_1, c_2, c_3 > 0$ are generic constants.

Therefore,

$$\begin{aligned}
&\mathbb{P}\left(\max_{j,k}\frac{1}{n}\sum_{X_{ij}\in\mathcal{E}_n, X_{ik}\in\mathcal{E}_n}|\Delta_i(j,k)| > \frac{\varepsilon}{4}\right) \\
&= \mathbb{P}\left(\max_{j,k}\frac{1}{n}\sum_{X_{ij}\in\mathcal{E}_n, X_{ik}\in\mathcal{E}_n}|\Delta_i(j,k)| > \frac{\varepsilon}{4}, \max_{j,k}\sup_{t\in\mathcal{E}_n, s\in\mathcal{E}_n}|\Theta_{t,s}(j,k)| > \theta_1\right) \\
&\quad + \mathbb{P}\left(\max_{j,k}\frac{1}{n}\sum_{X_{ij}\in\mathcal{E}_n, X_{ik}\in\mathcal{E}_n}|\Delta_i(j,k)| > \frac{\varepsilon}{4}, \max_{j,k}\sup_{t\in\mathcal{E}_n, s\in\mathcal{E}_n}|\Theta_{t,s}(j,k)| \le \theta_1\right) \\
&\le \mathbb{P}\left(\max_{j,k}\sup_{t\in\mathcal{E}_n, s\in\mathcal{E}_n}|\Theta_{t,s}(j,k)| > \theta_1\right) + \mathbb{P}\left(\frac{1}{n}\sum_{i=1}^n \mathbf{1}_{\{X_{ij}\in\mathcal{E}_n, X_{ik}\in\mathcal{E}_n\}} > \frac{\varepsilon}{4\theta_1}\right) \\
&= \mathbb{P}\left(\max_{j,k}\sup_{t\in\mathcal{E}_n, s\in\mathcal{E}_n}|\Theta_{t,s}(j,k)| > \theta_1\right) + o(1).
\end{aligned}$$

Now, we analyze the first term

$$\begin{aligned}\mathbb{P}\left(\max_{j,k}\sup_{t\in\mathcal{E}_n,s\in\mathcal{E}_n}|\Theta_{t,s}(j,k)|>\theta_1\right) &\leq p^2\mathbb{P}\left(\sup_{t\in\mathcal{E}_n,s\in\mathcal{E}_n}|\Theta_{t,s}(j,k)|>\theta_1\right)\\ &= p^2\mathbb{P}\left(\sup_{t\in\mathcal{E}_n,s\in\mathcal{E}_n}|\widetilde{f}_j(t)\widetilde{f}_k(s)-f_j(t)f_k(s)|>\theta_1\right).\end{aligned}$$

By adding and subtracting terms $f_j(t)$ and $f_s(t)$, we have

$$\begin{aligned}&\mathbb{P}\left(\sup_{t\in\mathcal{E}_n,s\in\mathcal{E}_n}|\widetilde{f}_j(t)\widetilde{f}_k(s)-f_j(t)f_k(s)|>\theta_1\right)\\ &\qquad\leq \mathbb{P}\left(\sup_{t\in\mathcal{E}_n,s\in\mathcal{E}_n}|(\widetilde{f}_j(t)-f_j(t))(\widetilde{f}_k(s)-f_k(s))|>\frac{\theta_1}{3}\right)\\ &\qquad\quad+\mathbb{P}\left(\sup_{t\in\mathcal{E}_n,s\in\mathcal{E}_n}|(\widetilde{f}_j(t)-f_j(t))|\cdot|f_k(s)|>\frac{\theta_1}{3}\right)\\ &\qquad\quad+\mathbb{P}\left(\sup_{t\in\mathcal{E}_n,s\in\mathcal{E}_n}|(\widetilde{f}_k(s)-f_k(s))|\cdot|f_j(t)|>\frac{\theta_1}{3}\right).\end{aligned}$$

The first term can further be decomposed to be

$$\begin{aligned}&\mathbb{P}\left(\sup_{t\in\mathcal{E}_n,s\in\mathcal{E}_n}|(\widetilde{f}_j(t)-f_j(t))(\widetilde{f}_k(s)-f_k(s))|>\frac{\theta_1}{3}\right)\\ &\quad\leq \mathbb{P}\left(\sup_{t\in\mathcal{E}_n}|(\widetilde{f}_j(t)-f_j(t))|>\sqrt{\frac{\theta_1}{3}}\right)+\mathbb{P}\left(\sup_{s\in\mathcal{E}_n}|(\widetilde{f}_k(s)-f_k(s))|>\sqrt{\frac{\theta_1}{3}}\right).\end{aligned}$$

Also, from the definition of $\mathcal{E}_n$, we have

$$\sup_{t\in\mathcal{E}_n}|f_j(t)|=\sup_{t\in\mathcal{E}_n}\left|g_j^{-1}(t)\right|\leq\sqrt{M\log n}.$$

Since $\varepsilon\geq C_M\sqrt{\frac{\log p\log^2 n}{n^{1/2}}}$, we have

$$\frac{\theta_1}{3}=\frac{n^{\beta/2}\varepsilon}{24A\sqrt{\log n}}\geq\frac{C_M\sqrt{\log p\log^2 n}}{24A\sqrt{\log n}}=2(M+2)\log n.$$

This implies that

$$\sqrt{\frac{\theta_1}{3}}\geq\sqrt{2(M+2)\log n}\ \text{ and }\ \frac{\theta_1}{3\sqrt{M\log n}}\geq\sqrt{2(M+2)\log n}.$$

Then, from Lemma 16, we get

$$\mathbb{P}\left(\sup_{t\in\mathcal{E}_n}|(\widetilde{f}_j(t)-f_j(t))|>\sqrt{\frac{\theta_1}{3}}\right)=0$$

and

$$\mathbb{P}\left(\sup_{t\in\mathcal{E}_n,s\in\mathcal{E}_n}|(\widetilde{f}_j(t)-f_j(t))|\cdot|f_k(s)|>\frac{\theta_1}{3}\right)=0.$$

The claim of the lemma then follows directly. ■

Remark 18 *From the above analysis, we see that the data in the tails doesn't affect the rate. Using exactly the same argument, we can also show that*

$$\mathbb{P}\left(\max_{j,k}\frac{1}{n}\sum_{X_{ij}\in\mathcal{M}_n,X_{ik}\in\mathcal{E}_n}|\Delta_i(j,k)|>\frac{\varepsilon}{4}\right)=o(1).$$

Lemma 19 *On the event $\mathcal{A}_n$, let $\beta=1/2$ and $\varepsilon\geq C_M\sqrt{\frac{\log p\log^2 n}{n^{1/2}}}$. We have*

$$\mathbb{P}\left(\max_{j,k}\frac{1}{n}\sum_{X_{ij}\in\mathcal{M}_n,X_{ik}\in\mathcal{M}_n}|\Delta_i(j,k)|>\frac{\varepsilon}{4}\right)\leq 2\exp\left(2\log p-\frac{n^{1/2}\varepsilon^2}{1232\pi^2\log^2 n}\right)+2\exp\left(2\log p-\frac{n^{1/2}}{8\pi\log n}\right).$$

Proof We have

$$\begin{aligned}\mathbb{P}\left(\max_{j,k}\frac{1}{n}\sum_{X_{ij}\in\mathcal{M}_n,X_{ik}\in\mathcal{M}_n}|\Delta_i(j,k)|>\frac{\varepsilon}{4}\right)&\leq p^2\mathbb{P}\left(\sup_{t\in\mathcal{M}_n,s\in\mathcal{M}_n}|\widetilde{f}_j(t)\widetilde{f}_k(s)-f_j(t)f_k(s)|>\frac{\varepsilon}{4}\right)\\&\leq p^2\mathbb{P}\left(\sup_{t\in\mathcal{M}_n,s\in\mathcal{M}_n}|(\widetilde{f}_j(t)-f_j(t))(\widetilde{f}_k(s)-f_k(s))|>\frac{\varepsilon}{12}\right)\\&\quad+2p^2\mathbb{P}\left(\sup_{t\in\mathcal{M}_n,s\in\mathcal{M}_n}|(\widetilde{f}_j(t)-f_j(t))|\cdot|f_k(s)|>\frac{\varepsilon}{12}\right).\end{aligned}$$

Further, since

$$\sup_{t\in\mathcal{M}_n}|f_j(t)|=\sup_{t\in\mathcal{M}_n}\left|g_j^{-1}(t)\right|=\sqrt{\beta\log n}$$

and $\sup_{t\in\mathcal{M}_n,s\in\mathcal{M}_n}|(\widetilde{f}_j(t)-f_j(t))(\widetilde{f}_k(s)-f_k(s))|$ is of higher order than $\sup_{t\in\mathcal{M}_n,s\in\mathcal{M}_n}|(\widetilde{f}_j(t)-f_j(t))|\cdot|f_k(s)|$, we only need to analyze the term $\mathbb{P}\left(\sup_{t\in\mathcal{M}_n}|(\widetilde{f}_j(t)-f_j(t))|>\frac{\varepsilon}{12\sqrt{\beta\log n}}\right)$.

Since $\delta_n=\frac{1}{4n^{\beta/2}\sqrt{2\pi\beta\log n}}$, using Mill's inequality we have

$$2\delta_n=\frac{\phi(\sqrt{\beta\log n})}{2\sqrt{\beta\log n}}\leq 1-\Phi(\sqrt{\beta\log n}).$$

This implies that

$$1-\delta_n-\Phi(\sqrt{\beta\log n})\geq\delta_n>0.$$

Using Lemma 14, we have

$$p^2\mathbb{P}\left(\widehat{F}_j\left(g_j\left(\sqrt{\beta\log n}\right)\right)>1-\delta_n\right)\leq p^2\exp\left(-2n\delta_n^2\right)=\exp\left(2\log p-\frac{n^{1-\beta}}{(16\pi\beta\log n)}\right) \tag{15}$$

and

$$p^2\mathbb{P}\left(\widehat{F}_j\left(g_j\left(-\sqrt{\beta\log n}\right)\right)<\delta_n\right)\leq\exp\left(2\log p-\frac{n^{1-\beta}}{(16\pi\beta\log n)}\right). \tag{16}$$

Define an event $\mathcal{B}_n$ as

$$\mathcal{B}_n\equiv\left\{\delta_n\leq\widehat{F}_j\left(g_j\left(\sqrt{\beta\log n}\right)\right)\leq 1-\delta_n, j=1,\ldots,p\right\}.$$

From (15) and (16), it is easy to see that

$$\mathbb{P}\left(\mathcal{B}_n^c\right)\leq 2\exp\left(2\log p-\frac{n^{1/2}}{8\pi\log n}\right).$$

From the definition of $\widetilde{F}_j$, we have

$$\begin{aligned}
&p^2\mathbb{P}\left(\sup_{t\in\mathcal{M}_n}|\widetilde{f}_j(t)-f_j(t)|>\frac{\varepsilon}{12\sqrt{\beta\log n}}\right)\\
&\leq\quad p^2\mathbb{P}\left(\sup_{t\in\mathcal{M}_n}\left|\Phi^{-1}\left(\widetilde{F}_j(t)\right)-\Phi^{-1}\left(F_j(t)\right)\right|>\frac{\varepsilon}{12\sqrt{\beta\log n}},\mathcal{B}_n\right)+\mathbb{P}\left(\mathcal{B}_n^c\right).\\
&\leq\quad p^2\mathbb{P}\left(\sup_{t\in\mathcal{M}_n}\left|\Phi^{-1}\left(\widehat{F}_j(t)\right)-\Phi^{-1}\left(F_j(t)\right)\right|>\frac{\varepsilon}{12\sqrt{\beta\log n}}\right)+2\exp\left(2\log p-\frac{n^{1/2}}{8\pi\log n}\right).
\end{aligned}$$

Define

$$T_{1n}\equiv\max\left\{F_j\left(g_j\left(\sqrt{\beta\log n}\right)\right),1-\delta_n\right\}\ \text{ and }\ T_{2n}\equiv 1-\min\left\{F_j\left(g_j\left(-\sqrt{\beta\log n}\right)\right),\delta_n\right\}.$$

From Equation (12) and the fact that $1-\delta_n\geq\Phi\left(\sqrt{\beta\log n}\right)$, we have that

$$T_{1n}=T_{2n}=1-\delta_n.$$

Thus, by the mean value theorem,

$$\begin{aligned}
&\mathbb{P}\left(\sup_{t\in\mathcal{M}_n}\left|\Phi^{-1}\left(\widehat{F}_j(t)\right)-\Phi^{-1}\left(F_j(t)\right)\right|>\frac{\varepsilon}{12\sqrt{\beta\log n}}\right)\\
&\leq\quad\mathbb{P}\left((\Phi^{-1})'\left(\max\left\{T_{1n},T_{2n}\right\}\right)\sup_{t\in\mathcal{M}_n}\left|\widehat{F}_j(t)-F_j(t)\right|>\frac{\varepsilon}{12\sqrt{\beta\log n}}\right)\\
&=\quad\mathbb{P}\left((\Phi^{-1})'\left(1-\delta_n\right)\sup_{t\in\mathcal{M}_n}\left|\widehat{F}_j(t)-F_j(t)\right|>\frac{\varepsilon}{12\sqrt{\beta\log n}}\right).
\end{aligned}$$

Finally, using the Dvoretzky-Kiefer-Wolfowitz inequality,

$$\begin{aligned}
&\mathbb{P}\left(\sup_{t\in\mathcal{M}_n}\left|\Phi^{-1}\left(\widehat{F}_j(t)\right)-\Phi^{-1}\left(F_j(t)\right)\right|>\frac{\varepsilon}{12\sqrt{\beta\log n}}\right)\\
&\qquad\leq\ \mathbb{P}\left(\sup_{t\in\mathcal{M}_n}\left|\widehat{F}_j(t)-F_j(t)\right|>\frac{\varepsilon}{(\Phi^{-1})'(1-\delta_n)\,12\sqrt{\beta\log n}}\right)\\
&\qquad\leq\ 2\exp\left(-2\frac{n\varepsilon^2}{144\beta\log n\left[(\Phi^{-1})'(1-\delta_n)\right]^2}\right).
\end{aligned}$$

Furthermore, by Lemma 11,

$$(\Phi^{-1})'(1-\delta_n)=\frac{1}{\phi(\Phi^{-1}(1-\delta_n))}\leq\frac{1}{\phi\left(\sqrt{2\log\frac{1}{\delta_n}}\right)}=\sqrt{2\pi}\left(\frac{1}{\delta_n}\right)=8\pi n^{\beta/2}\sqrt{\beta\log n}.$$

This implies that

$$p^2\mathbb{P}\left(\sup_{t\in\mathcal{M}_n}\left|\Phi^{-1}\left(\widehat{F}_j(t)\right)-\Phi^{-1}\left(F_j(t)\right)\right|>\frac{\varepsilon}{12\sqrt{\beta\log n}}\right)\leq 2\exp\left(2\log p-\frac{n^{1/2}\varepsilon^2}{1232\pi^2\log^2 n}\right).$$

In summary, we have

$$\mathbb{P}\left(\max_{j,k}\frac{1}{n}\sum_{X_{ij}\in\mathcal{M}_n,X_{ik}\in\mathcal{E}_n}|\Delta_i(j,k)|>\frac{\varepsilon}{4}\right)\leq 2\exp\left(2\log p-\frac{n^{1/2}\varepsilon^2}{1232\pi^2\log^2 n}\right)+2\exp\left(2\log p-\frac{n^{1/2}}{8\pi\log n}\right)$$

This finish the proof. ■

The conclusion of Theorem 4 follows from Lemma 17 and Lemma 19.

7.2 Proof of Theorem 8

Proof First note that the population and sample risks are

$$\begin{aligned}
R(f,\Omega) &= \frac{1}{2}\left\{\mathrm{tr}\left[\Omega\mathbb{E}(f(X)f(X)^T\right]-\log|\Omega|-p\log(2\pi)\right\}\\
\widehat{R}(f,\Omega) &= \frac{1}{2}\left\{\mathrm{tr}\left[\Omega S_n(f)\right]-\log|\Omega|-p\log(2\pi)\right\}.
\end{aligned}$$

Therefore, for all $(f,\Omega)\in\mathcal{M}_n^p\oplus\mathcal{C}_n$, we have

$$\begin{aligned}
|\widehat{R}(f,\Omega)-R(f,\Omega)| &= \frac{1}{2}\left|\mathrm{tr}\left[\Omega\left(\mathbb{E}[ff^T]-S_n(f)\right)\right]\right|\\
&\leq \frac{1}{2}\|\Omega\|_1\max_{jk}\sup_{f_j,f_k\in\mathcal{M}_n}|\mathbb{E}(f_j(X_j)f_k(X_k)-S_n(f)_{jk}|\\
&\leq \frac{L_n}{2}\max_{jk}\sup_{f_j,f_k\in\mathcal{M}_n}|\mathbb{E}(f_j(X_j)f_k(X_k)-S_n(f)_{jk}|.
\end{aligned}$$

Now, if $\mathcal{F}$ is a class of functions, we have

$$\mathbb{E}\left(\sup_{g\in\mathcal{F}}|\widehat{\mu}(g)-\mu(g)|\right)\leq\frac{CJ_{[]}(\|F\|_\infty,\mathcal{F})}{\sqrt{n}} \tag{17}$$

for some $C>0$, where $F(x)=\sup_{g\in cF}|g(x)|, \mu(g)=\mathbb{E}(g(X))$ and $\widehat{\mu}(g)=n^{-1}\sum_{i=1}^n g(X_i)$ (see Corollary 19.35 of van der Vaart 1998). Here the bracketing integral is defined to be

$$J_{[]}(\delta,\mathcal{F})=\int_0^\delta\sqrt{\log N_{[]}(u,\mathcal{F})}\,du$$

where $\log N_{[]}(\varepsilon,\mathcal{F})$ is the bracketing entropy. For the class of one dimensional, bounded and monotone functions, the bracketing entropy satisfies

$$\log N_{[]}(\varepsilon,\mathcal{M})\leq K\left(\frac{1}{\varepsilon}\right)$$

for some $K>0$ (van der Vaart and Wellner, 1996).

Now, let $\mathcal{P}_{n,p}$ be the class of all functions of the form $m(x)=f_j(x_j)f_k(x_k)$ for $j,k\in\{1,\ldots,p\}$, where $f_j\in\mathcal{M}_n$ for each j. Then the bracketing entropy satisfies

$$\log N_{[]}(C\sqrt{\log n},\mathcal{P}_{n,p})\leq 2\log p+K\left(\frac{1}{\varepsilon}\right)$$

and the bracketing integral satisfies $J_{[]}(C\sqrt{\log n},\mathcal{P}_{n,p})=O(\sqrt{\log n\log p})$. It follows from (17) and Markov's inequality that

$$\max_{jk}\sup_{f_j,f_k\in\mathcal{M}_n}|S_n(f)_{jk}-\mathbb{E}(f_j(X_j)f_k(X_k)|=O_P\left(\sqrt{\frac{\log n\log p}{n}}\right)=O_P\left(\sqrt{\frac{\log n}{n^{1-\xi}}}\right).$$

Therefore,

$$\sup_{(f,\Omega)\in\mathcal{M}_n^p\oplus\mathcal{C}_n}|\widehat{R}(f,\Omega)-R(f,\Omega)|=O_P\left(\frac{L_n\sqrt{\log n}}{n^{(1-\xi)/2}}\right).$$

As a consequence, we have

$$\begin{aligned}
R(f^*,\Omega^*) &\leq R(\widetilde{f}_n,\widehat{\Omega}_n)\\
&\leq \widehat{R}(\widetilde{f}_n,\widehat{\Omega}_n)+O_P\left(\frac{L_n\sqrt{\log n}}{n^{(1-\xi)/2}}\right)\\
&\leq \widehat{R}(f^*,\Omega^*)+O_P\left(\frac{L_n\sqrt{\log n}}{n^{(1-\xi)/2}}\right)\\
&\leq R(f^*,\Omega^*)+O_P\left(\frac{L_n\sqrt{\log n}}{n^{(1-\xi)/2}}\right)
\end{aligned}$$

and the conclusion follows. ■

8. Concluding Remarks

In this paper we have introduced the nonparanormal, a type of Gaussian copula with nonparametric marginals that is suitable for estimating high dimensional undirected graphs. The nonparanormal can be viewed as an extension of sparse additive models to the setting of graphical models. We proposed an estimator for the component functions that is based on thresholding the tails of the empirical distribution function at appropriate levels. A theoretical analysis was given to bound the difference between the sample covariance with respect to these estimated functions and the true sample covariance. This analysis was leveraged with the recent work of Ravikumar et al. (2009b) and Rothman et al. (2008) to obtain consistency results for the nonparanormal. Computationally, fitting a high dimensional nonparanormal is no more difficult than estimating a multivariate Gaussian, and indeed one can exploit existing software for the graphical lasso. Our experimental results indicate that the sparse nonparanormal can give very different results than a sparse Gaussian graphical model. This suggests that it may be a useful tool for relaxing the normality assumption, which is often made only for convenience.

Acknowledgments

We thank Zoubin Ghahramani, Michael Jordan, and the anonymous reviewers for helpful comments on this work. The research reported here was supported in part by NSF grant CCF-0625879 and a grant from Google.

References

Felix Abramovich, Yoav Benjamini, David L. Donoho, and Iain M. Johnstone. Adapting to unknown sparsity by controlling the false discovery rate. *The Annals of Statistics*, 34(2):584–653, 2006.

Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation. *Journal of Machine Learning Research*, 9:485–516, March 2008.

Tony Cai, Cun-Hui Zhang, and Harrison H. Zhou. Optimal rates of convergence for covariance matrix estimation. Technical report, Wharton School, Statistics Department, University of Pennsylvania, 2008.

Mathias Drton and Michael D. Perlman. Multiple testing and error control in Gaussian graphical model selection. *Statistical Science*, 22(3):430–449, 2007.

Mathias Drton and Michael D. Perlman. A SINful approach to Gaussian graphical model selection. *Journal of Statistical Planning and Inference*, 138(4):1179–1200, 2008.

Jerome H. Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2007.

Trevor Hastie and Robert Tibshirani. *Generalized additive models*. Chapman & Hall Ltd., 1999.

Chris A. J. Klaassen and Jon A. Wellner. Efficient estimation in the bivariate normal copula model: Normal margins are least-favorable. *Bernoulli*, 3(1):55–77, 1997.

Colin L. Mallows, editor. *The collected works of John W. Tukey. Volume VI: More mathematical, 1938–1984*. Wadsworth & Brooks/Cole, 1990.

Nicolai Meinshausen and Peter Bühlmann. High dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, 34:1436–1462, 2006.

Pradeep Ravikumar, Han Liu, John Lafferty, and Larry Wasserman. SpAM: Sparse additive models. In *Advances in Neural Information Processing Systems 20*, pages 1201–1208. MIT Press, Cambridge, MA, 2008.

Pradeep Ravikumar, John Lafferty, Han Liu, and Larry Wasserman. Sparse additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, 2009a. To appear.

Pradeep Ravikumar, Martin Wainwright, Garvesh Raskutti, and Bin Yu. Model selection in Gaussian graphical models: High-dimensional consistency of ℓ_1-regularized MLE. In *Advances in Neural Information Processing Systems 22*, Cambridge, MA, 2009b. MIT Press.

Adam J. Rothman, Peter J. Bickel, Elizaveta Levina, and Ji Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494–515, 2008.

Abe Sklar. Fonctions de répartition à *n* dimensions et leurs marges. *Publications de l'Institut de Statistique de L'Université de Paris 8*, pages 229–231, 1959.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B, Methodological*, 58:267–288, 1996.

Hideatsu Tsukahara. Semiparametric estimation in copula models. *Canadian Journal of Statistics*, 33:357–375, 2005.

Aad W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.

Aad W. van der Vaart and Jon A. Wellner. *Weak Convergence and Empirical Processes: With Applications to Statistics*. Springer-Verlag, 1996.

Anja Wille et al. Sparse Gaussian graphical modelling of the isoprenoid gene network in *Arabidopsis thaliana*. *Genome Biology*, 5:R92, 2004.

Ming Yuan and Yi Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

Journal of Machine Learning Research 10 (2009) 2329-2348 Submitted 3/08; Revised 1/09; Published 10/09

Computing Maximum Likelihood Estimates in Recursive Linear Models with Correlated Errors

Mathias Drton DRTON@UCHICAGO.EDU
Department of Statistics
University of Chicago
Chicago, IL 60637, USA

Michael Eichler M.EICHLER@MAASTRICHTUNIVERSITY.NL
Department of Quantitative Economics
Maastricht University
P.O. Box 616,
6200 MD Maastricht
The Netherlands

Thomas S. Richardson THOMASR@UW.EDU
Department of Statistics
University of Washington
Box 354322
Seattle, WA 98195-4322, USA

Editor: Michael I. Jordan

Abstract

In recursive linear models, the multivariate normal joint distribution of all variables exhibits a dependence structure induced by a recursive (or acyclic) system of linear structural equations. These linear models have a long tradition and appear in seemingly unrelated regressions, structural equation modelling, and approaches to causal inference. They are also related to Gaussian graphical models via a classical representation known as a path diagram. Despite the models' long history, a number of problems remain open. In this paper, we address the problem of computing maximum likelihood estimates in the subclass of 'bow-free' recursive linear models. The term 'bow-free' refers to the condition that the errors for variables i and j be uncorrelated if variable i occurs in the structural equation for variable j. We introduce a new algorithm, termed Residual Iterative Conditional Fitting (RICF), that can be implemented using only least squares computations. In contrast to existing algorithms, RICF has clear convergence properties and yields exact maximum likelihood estimates after the first iteration whenever the MLE is available in closed form.

Keywords: linear regression, maximum likelihood estimation, path diagram, structural equation model, recursive semi-Markov model, residual iterative conditional fitting

1. Introduction

A system of linear structural equations determines a linear model for a set of variables by dictating that, up to a random error term, each variable is equal to a linear combination of some of the remaining variables. Traditionally the errors are assumed to have a centered joint multivariate normal distribution. Presenting a formalism for simultaneously representing causal and statistical hypothe-

ses (Pearl, 2000; Spirtes et al., 2000), these normal linear models, which are also called *structural equation models*, are widely used in the social sciences (Bollen, 1989) and many other contexts.

In seminal work, Wright (1921, 1934) introduced *path diagrams*, which are useful graphical representations of structural equations. A path diagram is a graph with one vertex for each variable and directed and/or bi-directed edges. A directed edge $i \to j$ indicates that variable i appears as covariate in the equation for variable j. The directed edges are thus in correspondence with the *path coefficients*, that is, the coefficients appearing in the linear structural equations. A bi-directed edge $i \leftrightarrow j$ indicates correlation between the errors in the equations for variables i and j. Graphs of this kind are also considered by Shpitser and Pearl (2006), who refer to them as recursive semi-Markovian causal models.

1.1 A Motivating Example

We motivate the normal linear models analyzed here with the following example, which is adapted from a more complex longitudinal study considered in Robins (2008).

Consider a two-phase sequential intervention study examining the effect of exercise and diet on diabetes. In the first phase patients are randomly assigned to a number of hours of exercise per week (Ex) drawn from a log-normal distribution. At the end of this phase blood pressure (BP) levels are measured. In the second phase patients are randomly assigned to a strict calorie controlled diet that produces a change in body-mass index (ΔBMI). The assigned change in BMI, though still randomized, is drawn, by design, from a normal distribution with mean depending linearly on $X = \log(\text{Ex})$ and BP. The dependence here is due to practical and ethical considerations. Finally at the end of the second phase, triglyceride levels (Y) indicating diabetic status are measured.

A question of interest is whether or not there is an effect of X on the outcome Y that is not mediated through the dependence of ΔBMI on X and BP. In other words, if there had been no ethical or practical restrictions, and the assignment (ΔBMI) in the second phase was completely randomized and thus independent of BP and X, would there still be any dependence between X and Y? Note that due to underlying confounding factors such as life history and genetic background, we would expect to observe dependence between BP and Y even if the null hypothesis of no effect of X on Y was true and the second treatment (ΔBMI) was completely randomized.

Our model consists of two pieces. First, the design of the study dictates that

$$X = \alpha_0 + \varepsilon_X, \tag{1}$$

$$\Delta\text{BMI} = \gamma_0 + \gamma_1 X + \gamma_2 \,\text{BP} + \varepsilon_{\Delta\text{BMI}}, \tag{2}$$

where $\varepsilon_X \sim \mathcal{N}(0, \sigma_X^2)$ and $\varepsilon_{\Delta\text{BMI}} \sim \mathcal{N}(0, \sigma_{\Delta\text{BMI}}^2)$ are independent. This assignment model is complemented by a model describing how BP and Y respond to the prior treatments:

$$\text{BP} = \beta_0 + \beta_1 X + \varepsilon_{\text{BP}}, \tag{3}$$

$$Y = \delta_0 + \delta_1 X + \delta_2 \,\Delta\text{BMI} + \varepsilon_Y, \tag{4}$$

where $(\varepsilon_{\text{BP}}, \varepsilon_Y)^t$ are centered bivariate normal and independent of ε_X and $\varepsilon_{\Delta\text{BMI}}$. We denote the variances of ε_{BP} and ε_Y by σ_{BP}^2 and σ_Y^2, respectively, and write $\sigma_{\text{BP},Y}$ for the possibly non-zero covariance of ε_{BP} and ε_Y. Figure 1 shows the path diagram for this structural equation model.

Equations (1), (2) and (3) simply specify conditional expectations that can be estimated in regressions. However, this is not the case in general with (4). Instead,

$$\text{E}\,[Y \mid X, \Delta\text{BMI}] = \bar{\delta}_0 + \bar{\delta}_1 X + \bar{\delta}_2 \,\Delta\text{BMI}$$

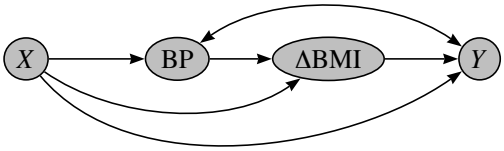


Figure 1: Path diagram illustrating a two-phase trial with two treatments (X and ΔBMI) and two responses (BP and Y). The treatment X is randomly assigned, and ΔBMI is randomized conditional on BP and X. The bi-directed edge indicates possible dependence due to unmeasured factors (genetic or environmental).

with

$$\bar{\delta}_1 = \delta_1 - \frac{\gamma_2 \sigma_{\mathrm{BP},Y}(\beta_1\gamma_2 + \gamma_1)}{\gamma_2^2\sigma_{\mathrm{BP}}^2 + \sigma_{\Delta\mathrm{BMI}}^2},$$

$$\bar{\delta}_2 = \delta_2 + \frac{\gamma_2 \sigma_{\mathrm{BP},Y}}{\gamma_2^2\sigma_{\mathrm{BP}}^2 + \sigma_{\Delta\mathrm{BMI}}^2},$$

and $\bar{\delta}_0 = \delta_0 + (\delta_1 - \bar{\delta}_1)\,\mathrm{E}[X] + (\delta_2 - \bar{\delta}_2)\,\mathrm{E}[\Delta\mathrm{BMI}]$. We see that δ_1 and δ_2 would have an interpretation as regression coefficients if: (i) the assignment of ΔBMI did not depend on BP (i.e., $\gamma_2 = 0$) and thus both treatments were completely randomized, or (ii) there were no dependence between ε_Y and $\varepsilon_{\mathrm{BP}}$ (i.e., $\sigma_{\mathrm{BP},Y} = 0$). Similarly, in $\mathrm{E}[Y \mid X, \mathrm{BP}, \Delta\mathrm{BMI}]$, the coefficient of ΔBMI is equal to δ_2 but the coefficient for X is $\delta_1 - \beta_1\sigma_{\mathrm{BP},Y}/\sigma_{\mathrm{BP}}^2$.

In this paper we consider likelihood-based methods for fitting a large class of structural equation models that includes the one given by (1)-(4) and can be used for consistent estimation of parameters such as δ_1. For alternative semi-parametric methods, see Robins (1999) and Gill and Robins (2001).

1.2 Challenges in Structural Equation Modelling

A number of mathematical and statistical problems arise in the normal linear models associated with systems of structural equations:

1. Different path diagrams may induce the same statistical model, that is, family of multivariate normal distributions. Such *model equivalence* occurs, for example, for the two path diagrams $1 \to 2$ and $1 \leftarrow 2$, which differ substantively by the direction of the cause-effect relationship. The two associated statistical models, however, are identical, both allowing for correlation between the two variables.

2. In many important special cases the path coefficient associated with a directed edge $i \to j$ has a population interpretation as a regression coefficient in a regression of j on a set of variables including i. However, as seen already in §1.1, this interpretation is not valid in general.

3. The parameters of the model may not be identifiable, so two different sets of parameter values may lead to the same population distribution; for an early review of this problem see Fisher (1966).

4. The set of parameterized covariance matrices may contain 'singularities' at which it cannot be approximated locally by a linear space. At 'singular' points, χ^2 and normal approximation to the distribution of likelihood ratio tests and maximum likelihood estimators (MLE) may not be valid; see, for instance, Drton (2009).

5. Iterative procedures are typically required for maximization of the likelihood function, which for some models can be multimodal (Drton and Richardson, 2004). Such multimodality typically occurs in small samples or under model misspecification.

The problems listed may arise in models without unobserved variables and become only more acute in latent variable models. They are challenging in full generality, but significant progress has been made in special cases such as recursive linear models with uncorrelated errors, which are also known as directed acyclic graph (DAG) models or 'Bayesian' networks (Lauritzen, 1996; Pearl, 1988). A normal DAG model is equivalent to a series of linear regressions, is always identified and has standard asymptotics. Under simple sample size conditions, the MLE exists almost surely and is a rational function of the data. Graphical modelling theory also solves problem 1 by characterizing all DAGs that induce the same statistical model (Andersson et al., 1997). For more recent progress on the general equivalence problem see Ali et al. (2009, 2005) and Zhang and Spirtes (2005).

1.3 Contribution of This Work

The requirement of uncorrelated errors may be overly restrictive in many settings. While arbitrary correlation patterns over the errors may yield rather ill-behaved statistical models, there are subclasses of models with correlated errors in which some of the nice properties of DAG models are preserved; compare McDonald (2002). In this paper we consider path diagrams in which there are no directed cycles and no 'double' edges of the form $i \leftrightarrows j$ (compare Def. 2 and 3). Since such double edges have been called 'bows', we call this class *bow-free acyclic path diagrams* (BAPs). An example of a BAP arose in our motivating example in §1.1; see Figure 1. While instrumental variable models, which are much studied in economics, contain bows, most models in other social sciences are based on BAPs. For instance, all path diagrams in Bollen (1989) are BAPs.

Bow-free acyclic path diagrams were also considered by Brito and Pearl (2002) who showed that the associated normal linear models are almost everywhere identifiable; see §2.2 for the definition. This result and other identification properties of BAP models are reviewed in Section 2. In Section 3 we give details on likelihood equations and Fisher-information of normal structural equation models. This sets the scene for our main contribution: the *Residual Iterative Conditional Fitting* (RICF) algorithm for maximization of the likelihood function of BAP models, which is presented in Section 4. Standard software for structural equation modelling currently employs general-purpose optimization routines for this task (Bollen, 1989, Appendix 4C). Many of these algorithms, however, neglect constraints of positive definiteness on the covariance matrix and suffer from convergence problems. According to Steiger (2001), failure to converge is 'not uncommon' and presents significant challenges to novice users of existing software. In contrast, our RICF algorithm produces positive definite covariance matrix estimates during all its iterations and has good convergence properties, as illustrated in the simulations in Section 5. Further discussion of RICF is provided in Section 6.

2. Normal Linear Models and Path Diagrams

Let $Y = (Y_i \mid i \in V) \in \mathbb{R}^V$ be a random vector, indexed by the finite set V, that follows a multivariate normal distribution $\mathcal{N}(0,\Sigma)$ with positive definite covariance matrix Σ. A zero mean vector is assumed merely to avoid notational overhead. The models we consider subsequently are induced by linear structural equations as follows.

2.1 Systems of Structural Equations and Path Diagrams

Let $\{\mathrm{pa}(i) \mid i \in V\}$ and $\{\mathrm{sp}(i) \mid i \in V\}$ be two families of index sets. For reasons explained below, we refer to these index sets as sets of parents and spouses, respectively. We require that $i \notin \mathrm{pa}(i) \cup \mathrm{sp}(i)$ for all $i \in V$; moreover, let the second family satisfy the symmetry condition that $j \in \mathrm{sp}(i)$ if and only if $i \in \mathrm{sp}(j)$. These two families determine a system of structural equations

$$Y_i = \textstyle\sum_{j \in \mathrm{pa}(i)} \beta_{ij} Y_j + \varepsilon_i, \quad i \in V, \tag{5}$$

whose zero-mean errors ε_i and ε_j are uncorrelated if $i \notin \mathrm{sp}(j)$, or equivalently, $j \notin \mathrm{sp}(i)$. The equations in (5) correspond to a *path diagram*, that is, a mixed graph G featuring both *directed* ($\rightarrow$) and *bi-directed* ($\leftrightarrow$) edges but no edges from a vertex i to itself (see Figures 1 and 2). The vertex set of G is the index set V, and G contains the edge $j \rightarrow i$ if and only if $j \in \mathrm{pa}(i)$, and the edge $j \leftrightarrow i$ if and only if $j \in \mathrm{sp}(i)$ (or equivalently, $i \in \mathrm{sp}(j)$). Subsequently, we exploit the path diagram representation of (5). If $i \rightarrow j$ is an edge in G, then we call i a *parent* of j, and if $i \leftrightarrow j$ is in G then i is referred to as a *spouse* of j. Thus, as remarked above, $\mathrm{pa}(i)$, $\mathrm{sp}(i)$ are, respectively, the sets of parents and spouses of i.

Let G be a path diagram and define $\mathbf{B}(G)$ to be the collection of all $V \times V$ matrices $B = (\beta_{ij})$ that satisfy

$$\beta_{ij} = 0 \quad \text{whenever } j \rightarrow i \text{ is not an edge in } G, \tag{6}$$

and are such that $I - B$ is invertible. Let $\mathbf{P}(V)$ be the cone of positive definite $V \times V$ matrices and $\mathbf{O}(G) \subseteq \mathbf{P}(V)$ the set of matrices $\Omega = (\omega_{ij}) \in \mathbf{P}(V)$ that satisfy

$$\omega_{ij} = 0 \quad \text{whenever } i \neq j \text{ and } j \leftrightarrow i \text{ is not in } G. \tag{7}$$

(Here and in the sequel, a symbol such as V denotes both a finite set and its cardinality.) The system (5) associated with the path diagram G can be written compactly as $Y = BY + \varepsilon$. If we assume that $B \in \mathbf{B}(G)$ and that the error covariance matrix $\mathrm{Var}(\varepsilon) = \Omega$ is in $\mathbf{O}(G)$, then (5) has a unique solution Y that is a multivariate normal random vector with covariance matrix $\Sigma = (I-B)^{-1}\Omega(I-B)^{-t}$. Here, I is the identity matrix and the superscript '$-t$' stands for transposition and inversion.

The above considerations lead to the following definition of a linear model associated with a path diagram (or equivalently, a system of structural equations).

Definition 1 *The normal linear model $\mathbf{N}(G)$ associated with a path diagram G is the family of multivariate normal distributions $\mathcal{N}(0,\Sigma)$ with covariance matrix in the set $\mathbf{P}(G) = \left\{(I-B)^{-1}\Omega(I-B)^{-t} \mid B \in \mathbf{B}(G), \Omega \in \mathbf{O}(G)\right\}$. We call the map $\Phi_G : \mathbf{B}(G) \times \mathbf{O}(G) \rightarrow \mathbf{P}(G)$ given by*

$$\Phi_G(B,\Omega) = (I-B)^{-1}\Omega(I-B)^{-t}$$

the parameterization map of $\mathbf{N}(G)$.

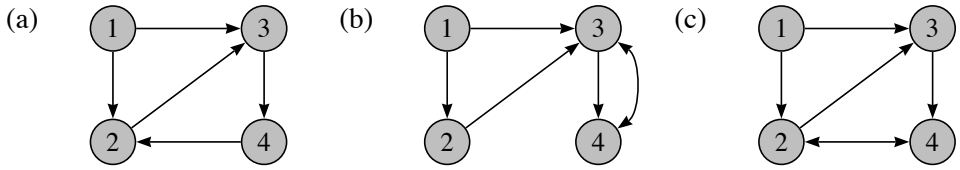


Figure 2: Path diagrams that are (a) cyclic, (b) acyclic but not bow-free, (c) acyclic and bow-free. Only path diagram (c) yields a curved exponential family.

Example 1 The path diagram G in Figure 2(a) depicts the equation system

$$\begin{aligned} Y_1 &= \varepsilon_1, & Y_2 &= \beta_{21}Y_1 + \beta_{24}Y_4 + \varepsilon_2, \\ Y_3 &= \beta_{31}Y_1 + \beta_{32}Y_2 + \varepsilon_3, & Y_4 &= \beta_{43}Y_3 + \varepsilon_4, \end{aligned}$$

where $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4$ are pairwise uncorrelated, that is, the matrices $\Omega \in \mathbf{O}(G)$ are diagonal. This system exhibits a circular covariate-response structure as the path diagram contains the directed cycle $2 \to 3 \to 4 \to 2$. This feedback loop is reflected in the fact that $\det(I-B) = 1 - \beta_{24}\beta_{43}\beta_{32}$ for $B \in \mathbf{B}(G)$. Therefore, the path coefficients need to satisfy $\beta_{24}\beta_{43}\beta_{32} \neq 1$ in order to lead to a positive definite covariance matrix in $\mathbf{P}(G)$. This example is considered in more detail in Drton (2009), where it is shown that the parameter space $\mathbf{P}(G)$ has singularities that lead to non-standard behavior of likelihood ratio tests.

The models considered in the remainder of this paper do not have any feedback loops, that is, they have the following structure.

Definition 2 *A path diagram G and its associated normal linear model $\mathbf{N}(G)$ are* recursive *or* acyclic *if G does not contain directed cycles, that is, there do not exist $i, i_1, \ldots, i_k \in V$ such that G features the edges $i \to i_1 \to i_2 \to \cdots \to i_k \to i$.*

We use the term *acyclic* rather than *recursive*, as some authors have used the term 'recursive' for path diagrams that are acyclic *and* contain no bi-directed edges. If G is acyclic, then the vertices in V can be ordered such that a matrix B that satisfies (6) is lower-triangular. It follows that

$$\det(I-B) = 1. \tag{8}$$

In particular, $I-B$ is invertible for any choice of the path coefficients β_{ij}, $j \to i$ in G, and the parameterization map Φ_G is a polynomial map.

2.2 Bow-free Acyclic Path Diagrams (BAPs)

The normal linear model $\mathbf{N}(G)$ associated with a path diagram G is *everywhere identifiable* if the parameterization map Φ_G is one-to-one, that is, for all $B_0 \in \mathbf{B}(G)$ and $\Omega_0 \in \mathbf{O}(G)$ it holds that

$$\Phi_G(B,\Omega) = \Phi_G(B_0,\Omega_0) \quad \Longrightarrow \quad B = B_0 \text{ and } \Omega = \Omega_0. \tag{9}$$

If there exists a Lebesgue null set $N_G \subseteq \mathbf{B}(G) \times \mathbf{O}(G)$ such that (9) holds for all $(B_0,\Omega_0) \notin N_G$, then we say that $\mathbf{N}(G)$ is *almost everywhere identifiable*.

Acyclic path diagrams may contain *bows*, that is, double edges $i \leftrightarrows j$. It is easy to see that normal linear models associated with path diagrams with bows are never everywhere identifiable. However, they may sometimes be almost everywhere identifiable as is the case for the next example. This example illustrates that almost everywhere identifiability is not enough to ensure regular behavior of statistical procedures.

Example 2 The path diagram G in Figure 2(b) features the bow $3 \leftrightarrows 4$. The associated normal linear model $\mathbf{N}(G)$ is also known as an instrumental variable model. The 9-dimensional parameter space $\mathbf{P}(G)$ is part of the hypersurface defined by the vanishing of the so-called *tetrad* $\sigma_{13}\sigma_{24} - \sigma_{14}\sigma_{23}$. It follows that the model $\mathbf{N}(G)$ lacks regularity because the tetrad hypersurface has singularities at points $\Sigma \in \mathbf{P}(G)$ with $\sigma_{13} = \sigma_{14} = \sigma_{23} = \sigma_{24} = 0$. These singularities occur if and only if $\beta_{31} = \beta_{32} = 0$, and correspond to points at which the identifiability property in (9) fails to hold. This lack of smoothness expresses itself statistically, for example, when testing the hypothesis $\beta_{31} = \beta_{32} = 0$ in model $\mathbf{N}(G)$. Using the techniques in Drton (2009), the likelihood ratio statistic for this problem can be shown to have non-standard behavior with a large-sample limiting distribution that is given by the larger of the two eigenvalues of a 2×2-Wishart matrix with 2 degrees of freedom and the identity matrix as scale parameter.

Definition 3 *A path diagram G and its associated normal linear model $\mathbf{N}(G)$ are* bow-free *if G contains at most one edge between any pair of vertices. If G is bow-free and acyclic, we call it a bow-free acyclic path diagram (BAP).*

As stressed in the introduction, BAPs are widespread in applications. Examples are shown in Figures 1, 2(c) and 6. Contrary to some path diagrams with bows, the normal linear models associated with BAPs are always at least almost everywhere identifiable.

Theorem 4 (Brito and Pearl, 2002) *If G is a BAP, then the normal linear model $\mathbf{N}(G)$ is almost everywhere identifiable.*

Many BAP models are in fact everywhere identifiable.

Theorem 5 (Richardson and Spirtes, 2002) *Suppose G is an ancestral BAP, that is, G does not contain an edge $i \leftrightarrow j$ such that there is a directed path $j \to i_1 \to \cdots \to i_k \to i$ that leads from vertex j to vertex i. Then the normal linear model $\mathbf{N}(G)$ is everywhere identifiable.*

The next example shows that the condition in Theorem 5 is sufficient but not necessary for identification. The characterization of the class of BAPs whose associated normal linear models are everywhere identifiable remains an open problem.

Example 3 The BAP G in Figure 2(c) is not ancestral because it contains the edges $4 \leftrightarrow 2 \to 3 \to 4$. Nevertheless, the associated normal linear model $\mathbf{N}(G)$ is everywhere identifiable, which can be shown by identifying the parameters in B and Ω row-by-row following the order $1 < 2 < 3 < 4$. It is noteworthy that the model $\mathbf{N}(G)$ in this example is not a Markov model, that is, a generic multivariate normal distribution in $\mathbf{N}(G)$ exhibits no conditional independence relations. Instead, the entries of covariance matrices $\Sigma = (\sigma_{ij}) \in \mathbf{P}(G)$ satisfy

$$(\sigma_{11}\sigma_{22} - \sigma_{12}^2)(\sigma_{14}\sigma_{33} - \sigma_{13}\sigma_{34}) = (\sigma_{13}\sigma_{24} - \sigma_{14}\sigma_{23})(\sigma_{12}\sigma_{13} - \sigma_{11}\sigma_{23}). \tag{10}$$

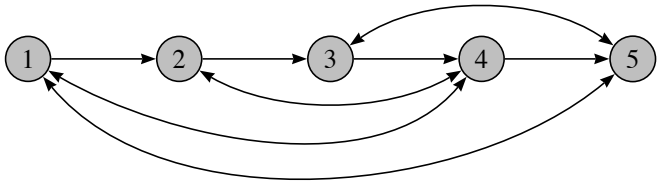


Figure 3: Bow-free acyclic path diagram whose associated normal linear model is almost, but not everywhere, identifiable. The model is not a curved exponential family.

The constraint in (10) has a nice interpretation. Let $(Y_1, \ldots, Y_4)$ have (positive definite) covariance matrix $\Sigma = (\sigma_{ij})$, and define $e_2 = Y_2 - \sigma_{21}/\sigma_{11} Y_1$ to be the residual in the regression of Y_2 on Y_1. Then (10) holds for Σ if and only if Y_1 and Y_4 are conditionally independent given e_2 and Y_3.

The above-stated Theorem 4 was proved in Brito and Pearl (2002), and an inspection of their proof reveals the following fact.

Lemma 6 *If the normal linear model $\mathbf{N}(G)$ associated with a BAP G is everywhere identifiable, then the (bijective) parameterization map Φ_G has an inverse that is a rational map with no pole on $\mathbf{P}(G)$.*

By (8), the parameterization map Φ_G for a BAP G is polynomial and thus smooth. If Φ_G^{-1} is rational and without pole, then the image of Φ_G, that is, $\mathbf{P}(G)$ is a smooth manifold (see, e.g., Edwards, 1994, II.4). This has an important consequence.

Corollary 7 *If the normal linear model $\mathbf{N}(G)$ associated with a BAP G is everywhere identifiable, then $\mathbf{N}(G)$ is a curved exponential family.*

The theory of curved exponential families is discussed by Kass and Vos (1997). It implies in particular that maximum likelihood estimators in curved exponential families are asymptotically normal, and that likelihood ratio statistics comparing two such families are asymptotically chi-square regardless of where in the null hypothesis a true parameter is located. Unfortunately, however, Lemma 6 and Corollary 7 do not hold for every BAP.

Example 4 The normal linear model associated with the BAP G in Figure 3 is not a curved exponential family. In this model the identifiability property in (9) breaks down if and only if (B, Ω) satisfy

$$\beta_{21}\omega_{14}\omega_{24} - \omega_2\omega_4 + \omega_{24}^2 = 0, \quad \beta_{32}\beta_{43}\omega_2 + \omega_{24} = 0.$$

It can be shown that the covariance matrices $\Phi_G(B, \Omega)$ associated with this set of parameters yield points at which the 13-dimensional set $\mathbf{P}(G)$ has more than 13 linearly independent tangent directions. Hence, $\mathbf{P}(G)$ is singular at these covariance matrices.

3. Likelihood Inference

Suppose a sample of size N is drawn from a multivariate normal distribution $\mathcal{N}(0,\Sigma)$ in the linear model $\mathbf{N}(G)$ associated with a BAP $G=(V,E)$. We group the observed random vectors as columns in the $V\times N$ matrix Y such that Y_{in} represents the observation of variable i on subject n. Having assumed a zero mean vector, we define the empirical covariance matrix as

$$S=(s_{ij})=\frac{1}{N}YY^t\in\mathbb{R}^{V\times V}.$$

Assuming $N\geq V$, the matrix S is positive definite with probability one. (As before, V here also denotes the cardinality of the set.) Models with unknown mean vector $\mu\in\mathbb{R}^V$ can be treated by estimating μ by the empirical mean vector and adjusting the empirical covariance matrix accordingly; $N\geq V+1$ then ensures almost sure positive definiteness of the empirical covariance matrix.

3.1 Likelihood Function and Likelihood Equations

Given observations Y with empirical covariance matrix S, the log-likelihood function $\ell:\mathbf{B}(G)\times\mathbf{O}(G)\to\mathbb{R}$ of the model $\mathbf{N}(G)$ takes the form

$$\ell(B,\Omega)=-\frac{N}{2}\log\det(\Omega)-\frac{N}{2}\mathrm{tr}\big[(I-B)^t\Omega^{-1}(I-B)S\big]. \tag{11}$$

Here we ignored an additive constant and used that $\det(I-B)=1$ if $B\in\mathbf{B}(G)$; compare (8). Let $\beta=(\beta_{ij}\mid i\in V,\ j\in\mathrm{pa}(i))$ and $\omega=(\omega_{ij}\mid i\leq j,\ j\in\mathrm{sp}(i)\text{ or }i=j)$ be the vectors of unconstrained elements in B and Ω. Let P and Q be the matrices with entries in $\{0,1\}$ that satisfy $\mathrm{vec}(B)=P\beta$ and $\mathrm{vec}(\Omega)=Q\omega$, respectively, where $\mathrm{vec}(A)$ refers to stacking of the columns of the matrix A. Taking the first derivatives of $\ell(B,\Omega)$ with respect to β and ω we obtain the likelihood equations.

Proposition 8 *The likelihood equations of the normal linear model $\mathbf{N}(G)$ associated with a BAP G can be written as*

$$P^t\,\mathrm{vec}\left(\Omega^{-1}(I-B)S\right)=P^t\,\mathrm{vec}\left(\Omega^{-1}S\right)-P^t\left(S\otimes\Omega^{-1}\right)P\beta=0, \tag{12}$$

$$Q^t\,\mathrm{vec}\left(\Omega^{-1}-\Omega^{-1}(I-B)S(I-B)^t\Omega^{-1}\right)=0, \tag{13}$$

where $\otimes$ denotes the Kronecker product.

In general, the likelihood equations need to be solved iteratively. One possible approach proceeds by alternately solving (12) and (13) for β and ω, respectively. For fixed ω, (12) is a linear equation in β and easily solved. For fixed β, (13) constitutes the likelihood equations of a multivariate normal covariance model for $\varepsilon=(I-B)Y$, which is specified by requiring that $\Omega_{ij}=0$ whenever the edge $i\leftrightarrow j$ is not in G. The solution of (13), with β fixed, requires, in general, another iterative method. As an alternative to this nesting of two iterative methods, we propose in Section 4 a method that solves (12) and (13) in joint updates of β and ω.

Remark 9 When proving their identifiability result for BAP models, Brito and Pearl (2002) gave an algorithm for recovering the parameters β and ω from a population covariance matrix. Applied to the empirical covariance matrix S, this algorithm produces consistent estimates $\tilde{\beta}$ and $\tilde{\omega}$. However, these are generally not the maximum likelihood estimators (MLE) and the error covariance matrix corresponding to $\tilde{\omega}$ may fail to be positive definite.

3.2 Fisher-Information

Large-sample confidence intervals for (β, ω) can be obtained by approximating the distribution of the MLE $(\hat{\beta}, \hat{\omega})$ by the normal distribution with mean vector (β, ω) and covariance matrix $\frac{1}{N} I(\beta, \omega)^{-1}$. Here, $I(\beta, \omega)$ denotes the Fisher-information, which, as shown in Appendix A, is of the following form.

Proposition 10 *The (expected) Fisher-information of the normal linear model* $\mathbf{N}(G)$ *associated with a BAP G is*

$$I(\beta, \omega) = \begin{pmatrix} P^t \left(\Sigma \otimes \Omega^{-1}\right) P & P^t \left[(I-B)^{-1} \otimes \Omega^{-1}\right] Q \\ Q^t \left[(I-B)^{-t} \otimes \Omega^{-1}\right] P & \frac{1}{2} Q^t \left(\Omega^{-1} \otimes \Omega^{-1}\right) Q \end{pmatrix}.$$

The Fisher-information in Proposition 10 need not be block-diagonal, in which case the estimation of the covariances ω affects the asymptotic variance of the MLE $\hat{\beta}$. However, this does not happen for *bi-directed chain graphs*, which form one of the model classes discussed by Wermuth and Cox (2004). A path diagram G is a bi-directed chain graph if its vertex set V can be partitioned into disjoint subsets $\tau_1, \ldots, \tau_T$, known as *chain components*, such that all edges in each subgraph G_{τ_t} are bi-directed and edges between two subsets $\tau_s \neq \tau_t$ are directed, pointing from τ_s to τ_t, if $s < t$. Since bi-directed chain graphs are ancestral graphs the associated normal linear models are everywhere identifiable.

Proposition 11 *For a BAP G, the following two statements are equivalent:*

(i) For all underlying covariance matrices $\Sigma \in \mathbf{P}(G)$*, the MLEs of the parameter vectors* β *and* ω *of the normal linear model* $\mathbf{N}(G)$ *are asymptotically independent.*

(ii) The path diagram G is a bi-directed chain graph.

A proof of Proposition 11 is given in Appendix A. This result is an instance of the asymptotic independence of mean and natural parameters in mixed parameterizations of exponential families (Barndorff-Nielsen, 1978).

4. Residual Iterative Conditional Fitting

We now present an algorithm for computing the MLE in the normal linear model $\mathbf{N}(G)$ associated with a BAP. The algorithm extends the *iterative conditional fitting* (ICF) procedure of Chaudhuri et al. (2007), which is for path diagrams with exclusively bi-directed edges.

Let $Y_i \in \mathbb{R}^N$ denote the i-th row of the observation matrix Y and $Y_{-i} = Y_{V \setminus \{i\}}$ the $(V \setminus \{i\}) \times N$ submatrix of Y. The ICF algorithm proceeds by repeatedly iterating through all vertices $i \in V$ and carrying out three steps: (i) fix the marginal distribution of Y_{-i}, (ii) fit the conditional distribution of Y_i given Y_{-i} under the constraints implied by the model $\mathbf{N}(G)$, and (iii) obtain a new estimate of Σ by combining the estimated conditional distribution $(Y_i \mid Y_{-i})$ with the fixed marginal distribution of Y_{-i}. The crucial point is then that for path diagrams containing only bi-directed edges, the problem of fitting the conditional distribution for $(Y_i \mid Y_{-i})$ under the constraints of the model can be rephrased as a least squares regression problem. Unfortunately, the consideration of the conditional distribution of $(Y_i \mid Y_{-i})$ is complicated for path diagrams that contain also directed edges. However, as we show below, the directed edges can be 'removed' by consideration of *residuals*, which here refers to estimates of the error terms $\varepsilon = (I - B)Y$. Since it is based on this idea, we give our new extended algorithm the name *Residual Iterative Conditional Fitting* (RICF).

4.1 The RICF Algorithm

The main building block of the new algorithm is the following decomposition of the log-likelihood function. We adopt the shorthand notation X_C for the $C \times N$ submatrix of a $D \times N$ matrix X, where $C \subseteq D$.

Theorem 12 *Let G be a BAP and $i \in V$. Let $\|x\|^2 = x^t x$ and define*

$$\omega_{ii.-i} = \omega_{ii} - \Omega_{i,-i}\Omega^{-1}_{-i,-i}\Omega_{-i,i} \tag{14}$$

to be the conditional variance of ε_i given ε_{-i}; recall that $\Omega^{-1}_{-i,-i} = (\Omega_{-i,-i})^{-1}$. Then the log-likelihood function $\ell(B,\Omega)$ of the model $\mathbf{N}(G)$ can be decomposed as

$$\begin{aligned}\ell(B,\Omega) = &-\frac{N}{2}\log\omega_{ii.-i} - \frac{1}{2\omega_{ii.-i}}\left\|Y_i - B_{i,\mathrm{pa}(i)}Y_{\mathrm{pa}(i)} - \Omega_{i,\mathrm{sp}(i)}(\Omega^{-1}_{-i,-i}\varepsilon_{-i})_{\mathrm{sp}(i)}\right\|^2 \\ &-\frac{N}{2}\log\det(\Omega_{-i,-i}) - \frac{1}{2}\mathrm{tr}\big(\Omega^{-1}_{-i,-i}\varepsilon_{-i}\varepsilon^t_{-i}\big).\end{aligned} \tag{15}$$

Proof Forming $\varepsilon = (I-B)Y$, we rewrite (11) as

$$\ell(B,\Omega) = -\frac{N}{2}\log\det(\Omega) - \frac{1}{2}\mathrm{tr}\big(\Omega^{-1}\varepsilon\varepsilon^t\big) =: \ell(\Omega \mid \varepsilon). \tag{16}$$

Using the inverse variance lemma (Whittaker, 1990, Prop. 5.7.3), we partition Ω^{-1} as

$$\begin{pmatrix}\omega_{ii} & \Omega_{i,-i}\\ \Omega_{-i,i} & \Omega_{-i,-i}\end{pmatrix}^{-1} = \begin{pmatrix}\omega^{-1}_{ii.-i} & -\omega^{-1}_{ii.-i}\Omega_{i,-i}\Omega^{-1}_{-i,-i}\\ -\Omega^{-1}_{-i,-i}\Omega_{-i,i}\omega^{-1}_{ii.-i} & \Omega^{-1}_{-i,-i} + \Omega^{-1}_{-i,-i}\Omega_{-i,i}\omega^{-1}_{ii.-i}\Omega_{i,-i}\Omega^{-1}_{-i,-i}\end{pmatrix}.$$

We obtain that the log-likelihood function in (16) equals

$$\begin{aligned}\ell(\Omega \mid \varepsilon) = &-\frac{N}{2}\log\omega_{ii.-i} - \frac{1}{2\omega_{ii.-i}}\left\|\varepsilon_i - \Omega_{i,-i}\Omega^{-1}_{-i,-i}\varepsilon_{-i}\right\|^2 \\ &-\frac{N}{2}\log\det(\Omega_{-i,-i}) - \frac{1}{2}\mathrm{tr}\big(\Omega^{-1}_{-i,-i}\varepsilon_{-i}\varepsilon^t_{-i}\big).\end{aligned}$$

By definition, $\varepsilon_i = Y_i - B_{i,\mathrm{pa}(i)}Y_{\mathrm{pa}(i)}$. Moreover, under the restrictions (7),

$$\Omega_{i,-i}\Omega^{-1}_{-i,-i}\varepsilon_{-i} = \Omega_{i,\mathrm{sp}(i)}(\Omega^{-1}_{-i,-i}\varepsilon_{-i})_{\mathrm{sp}(i)},$$

which yields the claimed decomposition. ■

The log-likelihood decomposition (15) is essentially based on the decomposition of the joint distribution of ε into the marginal distribution of ε_{-i} and the conditional distribution $(\varepsilon_i \mid \varepsilon_{-i})$. This leads to the idea of building an iterative algorithm whose steps are based on fixing the marginal distribution of ε_{-i} and estimating a conditional distribution. In order to fix the marginal distribution ε_{-i} we need to fix the submatrix $\Omega_{-i,-i}$ comprising all but the i-th row and column of Ω as well as the submatrix $B_{-i,V}$, which comprises all but the i-th row of B. With $\Omega_{-i,-i}$ and $B_{-i,V}$ fixed we can compute ε_{-i} as well as the *pseudo-variables*, defined by

$$Z_{-i} = \Omega^{-1}_{-i,-i}\varepsilon_{-i}. \tag{17}$$

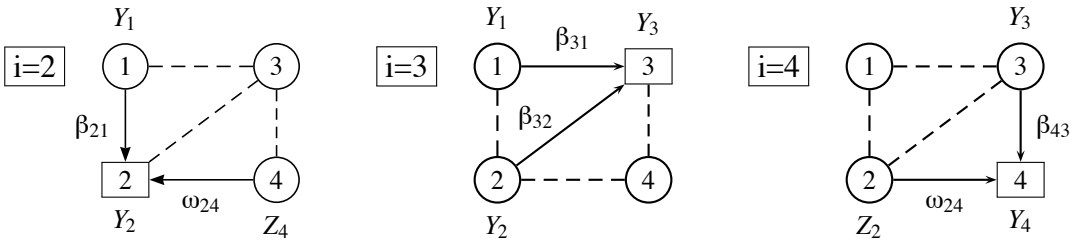


Figure 4: Illustration of the RICF update steps in Example 5. The structure of each least squares regression is indicated by directed edges pointing from the predictor variables to the response variable depicted by a square node. (See text for details.)

From (15), it now becomes apparent that, for fixed $\Omega_{-i,-i}$ and $B_{-i,V}$, the maximization of the log-likelihood function $\ell(B,\Omega)$ can be solved by maximizing the function

$$\big((\beta_{ij})_{j\in\mathrm{pa}(i)},(\omega_{ik})_{k\in\mathrm{sp}(i)},\omega_{ii.-i}\big)\mapsto -\frac{N}{2}\log\omega_{ii.-i}-\frac{1}{2\omega_{ii.-i}}\Big\|Y_i-\sum_{j\in\mathrm{pa}(i)}\beta_{ij}Y_j-\sum_{k\in\mathrm{sp}(i)}\omega_{ik}Z_k\Big\|^2 \quad (18)$$

over $\mathbb{R}^{\mathrm{pa}(i)}\times\mathbb{R}^{\mathrm{sp}(i)}\times(0,\infty)$. The maximizers of (18), however, are the least squares estimates in the regression of Y_i on both $(Y_j \mid j\in\mathrm{pa}(i))$ and $(Z_k \mid k\in\mathrm{sp}(i))$.

Employing the above observations, the *RICF algorithm* for computing the MLE $(\hat{B},\hat{\Omega})$ repeats the following steps for each $i\in V$:

1. Fix $\Omega_{-i,-i}$ and $B_{-i,V}$, and compute residuals ε_{-i} and pseudo-variables $Z_{\mathrm{sp}(i)}$;
2. Obtain least squares estimates of β_{ij}, $j\in\mathrm{pa}(i)$, ω_{ik}, $k\in\mathrm{sp}(i)$, and $\omega_{ii.-i}$ by regressing response variable Y_i on the covariates Y_j, $j\in\mathrm{pa}(i)$ and Z_k, $k\in\mathrm{sp}(i)$;
3. Compute an estimate of $\omega_{ii}=\omega_{ii.-i}+\Omega_{i,-i}\Omega^{-1}_{-i,-i}\Omega_{-i,i}$ using the new estimates and the fixed parameters; compare (14).

After steps 1 to 3, we move on to the next vertex in V. After the last vertex in V we return to consider the first vertex. The procedure is continued until convergence.

Example 5 For illustration of the regressions performed in RICF, we consider the normal linear model associated with the BAP G in Figure 2(c). The parameters to be estimated in this model are $\beta_{21},\beta_{31},\beta_{32},\beta_{43}$ and $\omega_{11},\omega_{22},\omega_{33},\omega_{44},\omega_{24}$.

Vertex 1 in Figure 2(c) has no parents or spouses, and its RICF update step consists of a trivial regression. In other words, the variance ω_{11} is the unconditional variance of Y_1 with MLE $\hat{\omega}_{11}=s_{11}$. For the remaining vertices, the corresponding RICF update steps are illustrated in Figure 4, where the response variable Y_i in the i-th update step is shown as a square node while the remaining variables are depicted as circles. Directed edges indicate variables acting as covariates in the least squares regression. These covariates are labelled according to whether the random variable Y_j, or the pseudo-variable Z_j defined in (17), is used in the regression. Note that since $\mathrm{sp}(3)=\emptyset$, repetition of steps 1-3 in §4.1 is required only for $i\in\{2,4\}$.

In RICF, the log-likelihood function $\ell(B,\Omega)$ from (11) is repeatedly maximized over sections in the parameter space defined by fixing the parameters $\Omega_{-i,-i}$, and $B_{-i,V}$. RICF thus is an iterative partial maximization algorithm and has the following convergence properties.

Theorem 13 *If G is a BAP and the empirical covariance matrix S is positive definite, then the following holds:*

(i) For any starting value $(\hat{B}^0,\hat{\Omega}^0) \in \mathbf{B}(G) \times \mathbf{O}(G)$, RICF constructs a sequence of estimates $(\hat{B}^s,\hat{\Omega}^s)_s$ in $\mathbf{B}(G) \times \mathbf{O}(G)$ whose accumulation points are local maxima or saddle points of the log-likelihood function $\ell(B,\Omega)$. Moreover, evaluating the log-likelihood function at different accumulation points yields the same value.

(ii) If the normal linear model $\mathbf{N}(G)$ is everywhere identifiable and the likelihood equations have only finitely many solutions then the sequence $(\hat{B}^s,\hat{\Omega}^s)_s$ converges to one of these solutions.

Proof Let $\ell(\Sigma)$ be the log-likelihood function for the model of all centered multivariate normal distributions on $\mathbb{R}^V$. If S is positive definite then the set C that comprises all positive definite matrices $\Sigma \in \mathbb{R}^{V\times V}$ at which $\ell(\Sigma) \geq \ell(\hat{B}^0,\hat{\Omega}^0)$ is compact. In particular, the log-likelihood function in (11) is bounded, and claim (i) can be derived from general results about iterative partial maximization algorithms; see for example, Drton and Eichler (2006). For claim (ii) note that if $\mathbf{N}(G)$ is everywhere identifiable, then the compact set C has compact preimage $\phi_G^{-1}(C)$ under the model parameterization map; recall Lemma 6. ■

Remark 14 If the normal linear model $\mathbf{N}(G)$ associated with a BAP G is not everywhere identifiable, then it is possible that a sequence of estimates $(\hat{B}^s,\hat{\Omega}^s)_s$ produced by RICF diverges and does not have any accumulation points. In these cases, however, the corresponding sequence of covariance matrices $\Phi_G(\hat{B}^s,\hat{\Omega}^s)_s$ still has at least one accumulation point because it ranges in the compact set C exhibited in the proof of Theorem 13. Divergence of $(\hat{B}^s,\hat{\Omega}^s)_s$ occurs in two instances in the simulations in §5; compare Table 1. In both cases, the sequence $\Phi_G(\hat{B}^s,\hat{\Omega}^s)_s$ converges to a positive definite covariance matrix.

4.2 Computational Savings in RICF

If G is a DAG, that is, an acyclic path diagram without bi-directed edges, then the MLE $(\hat{B},\hat{\Omega})$ in $\mathbf{N}(G)$ can be found in a finite number of regressions (e.g., Wermuth, 1980). However, we can also run RICF. Since in a DAG, $\mathrm{sp}(i) = \emptyset$ for all $i \in V$, step 2 of RICF regresses variable Y_i solely on its parents Y_j, $j \in \mathrm{pa}(i)$. Not involving pseudo-variables that could change from one iteration to the other, this regression remains the same throughout different iterations, and RICF converges in one step.

Similarly, for a general BAP G, if vertex $i \in V$ has no spouses, $\mathrm{sp}(i) = \emptyset$, then the MLE of $B_{i,\mathrm{pa}(i)}$ and ω_{ii} can be determined by a single iteration of the algorithm. In other words, RICF reveals these parameters as being estimable in closed form, namely as rational functions of the data. (This occurred for vertex $i = 3$ in Example 5.) It follows that, to estimate the remaining parameters, the iterations need only be continued over vertices j with $\mathrm{sp}(j) \neq \emptyset$.

For further computational savings note that $\Omega_{\mathrm{dis}(i),V\setminus(\mathrm{dis}(i)\cup\{i\})} = 0$, where $\mathrm{dis}(i) = \{j \mid j \leftrightarrow \cdots \leftrightarrow i, j \neq i\}$ is the district of $i \in V$. Hence, since $\mathrm{sp}(i) \subseteq \mathrm{dis}(i)$,

$$(\Omega^{-1}_{-i,-i}\varepsilon_{-i})_{\mathrm{sp}(i)} = (\Omega^{-1}_{\mathrm{dis}(i),\mathrm{dis}(i)}\varepsilon_{\mathrm{dis}(i)})_{\mathrm{sp}(i)};$$

see Koster (1999, Lemma 3.1.6) and Richardson and Spirtes (2002, Lemma 8.10). Since $\varepsilon_{\mathrm{dis}(i)} = Y_{\mathrm{dis}(i)} - B_{\mathrm{dis}(i),\mathrm{pa}(\mathrm{dis}(i))}Y_{\mathrm{pa}(\mathrm{dis}(i))}$, it follows that in the RICF update step for vertex i attention can be restricted to the variables in $\{i\} \cup \mathrm{pa}(i) \cup \mathrm{dis}(i) \cup \mathrm{pa}(\mathrm{dis}(i))$.

Finally, note that while the RICF algorithm is described in terms of the entire data matrix Y, the least squares estimates computed in its iterations are clearly functions of the empirical covariance matrix, which is a sufficient statistic.

5. Simulation Studies

In order to evaluate the performance of the RICF algorithm we consider two scenarios. First, we fit linear models based on randomly generated BAPs to gene expression data. This scenario is relevant for model selection tasks, and we compare RICF's performance in this problem to that of algorithms implemented in software for structural equation modelling. Second, we study how RICF behaves when it is used to fit larger models to data simulated from the respective model. In contrast to the first scenario, the second scenario involves models that generally fit the considered data well.

5.1 Gene Expression Data

We consider data on gene expression in *Arabidopsis thaliana* from Wille et al. (2004). We restrict attention to 13 genes that belong to one pathway: DXPS1-3, DXR, MCT, CMK, MECPS, HDS, HDR, IPPI1, GPPS, PPDS1-2. Data from $n = 118$ microarray experiments are available. We fit randomly generated BAP models to these data using RICF and two alternative methods.

The BAP models are generated as follows. For each of the 78 possible pairs of vertices $i < j$ in $V = \{1,\ldots,13\}$ we draw from a multinomial distribution to generate a possible edge. The probability for drawing the edge $i \to j$ is d, and the probability for drawing $i \leftrightarrow j$ is b so that with probability $1-d-b$ there is no edge between i and j. We then apply a random permutation to the vertices to obtain the final BAP. For each of twelve combinations (d,b) with $d = 0.05, 0.1, 0.2, 0.3$ and $b = 0.05, 0.1, 0.2$, we simulate 1000 BAPs. The expected number of edges thus varies between 7.8 and 39.

For fitting the simulated BAPs to the gene expression data, we implemented RICF in the statistical programming environment R (R Development Core Team, 2007). As alternatives, we consider the R package 'sem' (Fox, 2006) and the widely used software LISREL (Jöreskog and Sörbom, 1997) in its student version 8.7 for Linux (student versions are free but limited to 15 variables). Both these programs employ general purpose optimizers, for example, 'sem' makes a call to the R function 'nlm'.

Our simulation results are summarized in Table 1. Each row in the table corresponds to a choice of the edge probabilities d and b. The first three columns count how often, in 1000 simulations, the three considered fitting routines failed to converge. The starting values of LISREL and 'sem' were set according to program defaults, and RICF was started by setting $\hat{B}^{(0)}$ and $\hat{\Omega}^{(0)}$ equal to the MLE in the DAG model associated with the DAG obtained by removing all bi-directed edges from the considered BAP.

		No convergence			All	All	Running time		
d	b	RICF	LIS	SEM	converge	agree	RICF	LIS	SEM
0.05	0.05	0	36	47	941	940	0.03	0.02	1.15
	0.1	0	177	221	746	739	0.09	0.03	1.58
	0.2	0	499	599	347	333	0.21	0.04	2.71
0.1	0.05	0	32	36	951	949	0.04	0.03	1.58
	0.1	0	137	193	786	780	0.09	0.03	2.09
	0.2	0	440	610	364	354	0.25	0.04	3.43
0.2	0.05	0	19	39	958	954	0.05	0.03	2.67
	0.1	0	91	176	815	808	0.13	0.04	3.34
	0.2	1	326	520	461	452	0.33	0.05	5.03
0.3	0.05	0	16	38	960	957	0.06	0.04	4.04
	0.1	0	59	136	859	850	0.17	0.04	4.96
	0.2	1	225	471	519	490	0.40	0.06	6.97

Table 1: Fitting simulated BAPs to gene expression data using RICF, LISREL and 'sem'. Each row is based on 1000 simulations. Running time is average CPU time (in sec.) for the cases in which all three algorithms converged. (See text for details.)

LISREL and 'sem' failed to converge for a rather large number of models. The LISREL output explained why convergence failed, and virtually all failures were due to the optimizer converging to matrices that were not positive definite. The remedy would be to try new starting values but doing this successfully in an automated fashion is a challenging problem in itself. For RICF convergence failure arose in only two cases. In both cases the RICF estimates $(\hat{B}, \hat{\Omega})$ had some diverging entries. Despite the divergence in (B, Ω)-space, the sequence of associated covariance matrices $\Phi_G(\hat{B}, \hat{\Omega})$ computed by RICF converged, albeit very slowly. Recall that this phenomenon is possible in models that are almost, but not everywhere, identifiable (Remark 14). In these examples LISREL produced similarly divergent sequences with approximately the same likelihood, and 'sem' reported convergence in one case but gave an estimate whose likelihood was nearly 40 points smaller than the intermediate estimates computed by LISREL and RICF.

The columns labelled 'All converge' and 'All agree' in Table 1 show how often all methods converged, and when this occurred, how often the three computed maxima of the log-likelihood function were the same up to one decimal place. Since all methods are for local maximization, substantial disagreements in the computed maxima can occur if the likelihood function is multimodal.

Finally, the last three columns give average CPU time use for the cases in which all three algorithms converge. These are quoted to show that RICF is competitive in terms of computational efficiency, but for the following reasons the precise times should not be used for a formal comparison. On the one hand, LISREL is fast because it is compiled code. This is not the case for the R-based 'sem' and RICF. On the other hand, the fitting routines in LISREL and 'sem' not only compute the MLE but also produce various other derived quantities of interest. This is in contrast to our RICF routine, which only computes the MLE.

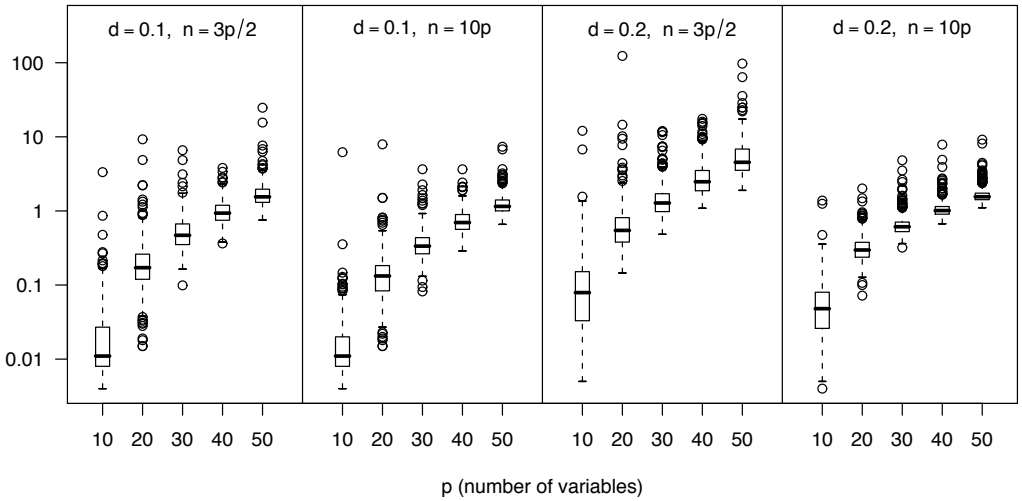


Figure 5: Boxplots of CPU times (in sec. on $\log_{10}$-scale) used by RICF when fitting BAP models to simulated data. Each boxplot summarizes 500 simulations. The number of variables is denoted by p, the sample size is n, and the parameter d determines the expected number of edges of the simulated BAPs (see text for details).

5.2 Simulated Data

In order to demonstrate how RICF behaves when fitting larger models we use the algorithm on simulated data. We consider different choices for the number of variables p and generate random BAPs according to the procedure used in §5.1. We limit ourselves to two different settings for the expected number of edges, choosing $d = 0.1$ or $d = 0.2$ and setting $b = d/2$ in each case. For each BAP G, we simulate a covariance matrix $\Sigma = (I-B)^{-1}\Omega(I-B)^{-t} \in \mathbf{P}(G)$ as follows. The free entries in $B \in \mathbf{B}(G)$ and the free off-diagonal entries in $\Omega \in \mathbf{O}(G)$ are drawn from a $\mathcal{N}(0,1)$ distribution. The diagonal entries ω_{ii} are obtained by adding a draw from a χ^2_1-distribution to the sum of the absolute values of the off-diagonal entries in the i-th row of Ω. This makes Ω diagonally dominant and thus positive definite. Finally, we draw a sample of size n from the resulting multivariate normal distribution $\mathbf{N}(G)$. For each distribution two cases, namely $n = 3p/2$ and $n = 10p$, are considered to illustrate sample size effects. For each combination of p, d and n, we simulate 500 BAPs and associated data sets.

Figure 5 summarizes the results of our simulations in boxplots. As could be expected, the running time for RICF increases with the number of variables p and the expected number of edges in the BAP (determined by d). Moreover, the running time decreases for increased sample size n, which is plausible because the empirical covariance matrix of a larger sample tends to be closer to the underlying parameter space $\mathbf{P}(G)$. The boxplots show that even with $p = 50$ variables the majority of the RICF computations terminate within a few seconds. However, there are also a number of computations in which the running time is considerably longer, though still under two

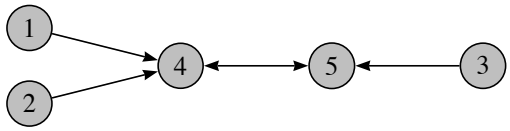


Figure 6: Path diagram for seemingly unrelated regressions.

minutes. This occurs in particular for the denser case with smaller sample size ($d = 0.2$ and $n = 3p/2$).

6. Discussion

As mentioned in the introduction, normal linear models associated with path diagrams are employed in many applied disciplines. The models, also known as structural equation models, have a long tradition but remain of current interest in particular due to the more recent developments in causal inference; compare, for example, Pearl (2000) and Spirtes et al. (2000). Despite their long tradition, however, many mathematical, statistical and computational problems about these models remain open.

The new contribution of this paper is the Residual Iterative Conditional Fitting (RICF) algorithm for maximum likelihood estimation in BAP models. Software for computation of MLEs in structural equation models often employs optimization methods that are not designed to deal with positive definiteness constraints on covariance matrices. This can be seen in particular in Table 1 which shows that two available programs, LISREL (Jöreskog and Sörbom, 1997) and the R package 'sem' (Fox, 2006), fail to converge in a rather large number of problems. This is in line with previous experience by other authors; see, for example, Steiger (2001). Our new RICF algorithm, on the other hand, does not suffer from these problems. It has clear convergence properties (Theorem 13) and can handle problems with several tens of variables (see Figure 5). In addition, RICF has the desirable feature that it estimates parameters in closed form (in a single cycle of its iterations) if this is possible. If applied to a model based on a directed acyclic graph (DAG), the algorithm converges in a single cycle and performs exactly the regressions commonly used for fitting multivariate normal DAG models. This feature and the fact that RICF can be implemented using nothing but least squares computations make it an attractive alternative to less specialized optimization methods.

In another special case, namely seemingly unrelated regressions, RICF reduces to the algorithm of Telser (1964). A path diagram representing seemingly unrelated regressions is shown in Figure 6. The variables Y_1, Y_2 and Y_3 are then commonly thought of as covariates. Since they have no spouses, the MLEs of the variances ω_{11}, ω_{22} and ω_{33} are equal to the empirical variances s_{11}, s_{22} and s_{33}. For the remaining variables Y_i, $i = 4, 5$, RICF performs regressions on both the "covariates" $Y_{\mathrm{pa}(i)}$ and the residual ε_j, $j \in \{4,5\} \setminus \{i\}$. These are precisely the steps performed by Telser.

Existing structural equation modelling software also fits models with latent variables, whereas the RICF algorithm applies only to BAP models without latent variables. However, RICF could be used to implement the M-step in the EM algorithm (Kiiveri, 1987) in order to fit latent variable models. This EM-RICF approach would yield an algorithm with theoretical convergence properties.

Finally, we emphasize that the RICF algorithm is determined by the path diagram. However, different path diagrams may induce the same statistical model; recall point (1) in §1.2 in the intro-

duction. This model equivalence of path diagrams may be exploited to find a diagram for which the running time of RICF is short. For example, for every BAP that is equivalent to a DAG model, parameter estimates could be computed in closed form and hence in finitely many steps. Relevant graphical constructions for this problem are described in Drton and Richardson (2008) and Ali et al. (2005).

Acknowledgments

This work was supported by the U.S. National Science Foundation (DMS-0505612, DMS-0505865, DMS-0746265); the Institute for Mathematics and its Applications; the U.S. National Institutes for Health (R01-HG2362-3, R01-AI032475).

Appendix A. Proofs

Proof [Proof of Proposition 10] Let β and ω be the vectors of unconstrained elements in B and Ω, respectively. The second derivatives of the log-likelihood function with respect to β and ω are:

$$\frac{\partial^2 \ell(B,\Omega)}{\partial\beta\,\partial\beta^t} = -N \cdot P^t \big(S \otimes \Omega^{-1}\big) P, \tag{19}$$

$$\frac{\partial^2 \ell(B,\Omega)}{\partial\beta\,\partial\omega^t} = -N \cdot P^t \big[S(I-B)^t\Omega^{-1} \otimes \Omega^{-1}\big] Q, \tag{20}$$

$$\begin{aligned}\frac{\partial^2 \ell(B,\Omega)}{\partial\omega\,\partial\omega^t} = -\frac{N}{2} Q^t \big\{ &\big[\Omega^{-1} \otimes \Omega^{-1}(I-B)S(I-B)^t\Omega^{-1}\big] \\ &+ \big[\Omega^{-1}(I-B)S(I-B)^t\Omega^{-1} \otimes \Omega^{-1}\big]\big\} Q.\end{aligned} \tag{21}$$

Replacing S by $\mathrm{E}[S] = (I-B)^{-1}\Omega(I-B)^{-t}$ in (19)-(21) yields the claim. ■

Proof [Proof of Proposition 11] If G is a bi-directed chain graph, then the submatrix $B_{\tau_t,\tau_t} = 0$ for all t, while for $s \neq t$ we have $\Omega_{\tau_s,\tau_t} = 0$. In this case the second derivative of the log-likelihood function with respect to β_{ij} and ω_{kl} is equal to $\partial^2\ell(B,\Omega)/\partial\beta_{ij}\,\partial\omega_{kl} = [(I-B)^{-1}]_{jl}\,(\Omega^{-1})_{ik}$. Now $[(I-B)^{-1}]_{jl}$ may only be non-zero if $j = l$ or l is an ancestor of j, that is, if there exists a directed path $l \to j_1 \to \cdots \to j_m \to j$ in G. On the other hand, $(\Omega^{-1})_{ik} = 0$ whenever i and k are not in the same chain component. Therefore, the second derivative in (20) is equal to zero.

Conversely, it follows that the second derivative in (20) vanishes for all parameters only if the graph belongs to the class of bi-directed chain graphs. ■

References

R. A. Ali, T. S. Richardson, P. Spirtes, and J. Zhang. Towards characterizing Markov equivalence classes for directed acyclic graphs with latent variables. In F. Bacchus and T. Jaakkola, editors, *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 10–17, Corvallis, Oregon, 2005. AUAI Press.

R. A. Ali, T. S. Richardson, and P. Spirtes. Markov equivalence for ancestral graphs. *Ann. Statist.*, 37:2808–2837, 2009.

S. A. Andersson, D. Madigan, and M. D. Perlman. A characterization of Markov equivalence classes for acyclic digraphs. *Ann. Statist.*, 25:505–541, 1997.

O. Barndorff-Nielsen. *Information and Exponential Families in Statistical Theory*. John Wiley & Sons Ltd., Chichester, 1978.

K. A. Bollen. *Structural Equations with Latent Variables*. Wiley, New York, 1989.

C. Brito and J. Pearl. A new identification condition for recursive models with correlated errors. *Struct. Equ. Model.*, 9:459–474, 2002.

S. Chaudhuri, M. Drton, and T. S. Richardson. Estimation of a covariance matrix with zeros. *Biometrika*, 94:199–216, 2007.

M. Drton. Likelihood ratio tests and singularities. *Ann. Statist.*, 37(2):979–1012, 2009.

M. Drton and M. Eichler. Maximum likelihood estimation in Gaussian chain graph models under the alternative Markov property. *Scand. J. Statist.*, 33:247–257, 2006.

M. Drton and T. S. Richardson. Multimodality of the likelihood in the bivariate seemingly unrelated regressions model. *Biometrika*, 91:383–392, 2004.

M. Drton and T. S. Richardson. Graphical methods for efficient likelihood inference in Gaussian covariance models. *J. Mach. Learn. Res.*, 9:893–914, 2008.

C. H. Edwards. *Advanced Calculus of Several Variables*. Dover Publications, New York, 1994.

F. M. Fisher. *The Identification Problem in Econometrics*. McGraw-Hill, New York, 1966.

J. Fox. Structural equation modeling with the sem package in R. *Struct. Equ. Model.*, 13:465–486, 2006.

R. D. Gill and J. M. Robins. Causal inference for complex longitudinal data: The continuous case. *Ann. Statist.*, 29(6):1785–1811, 2001.

K. Jöreskog and D. Sörbom. *LISREL 8: User's Reference Guide*. Scientific Software International, Lincolnwood, IL, 1997.

R. E. Kass and P. W. Vos. *Geometrical Foundations of Asymptotic Inference*. Wiley, New York, 1997.

H. T. Kiiveri. An incomplete data approach to the analysis of covariance structures. *Psychometrika*, 52:539–554, 1987.

J. T. A. Koster. *Linear structural equations and graphical models*. The Fields Institute, Lecture notes, Toronto, 1999.

S. L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, UK, 1996.

R. P. McDonald. What can we learn from the path equations? Identifiability, Constraints, Equivalence. *Psychometrika*, 67:225–249, 2002.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA, 1988.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, UK, 2000.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007.

T. S. Richardson and P. Spirtes. Ancestral graph Markov models. *Ann. Statist.*, 30:962–1030, 2002.

J. M. Robins. Testing and estimation of direct effects by reparameterizing directed acyclic graphs with structural nested models. In C. Glymour and G. Cooper, editors, *Computation, Causation, and Discovery*, pages 349–405. MIT Press, Cambridge, MA, 1999.

J. M. Robins. Causal models for estimating the effects of weight gain on mortality. *International Journal of Obesity*, 32:S15–S41, 2008.

I. Shpitser and J. Pearl. Identification of joint interventional distributions in recursive semi-Markovian causal models. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*. AAAI Press, 2006.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, Cambridge, MA, second edition, 2000.

J. H. Steiger. Driving fast in reverse. *J. Amer. Statist. Assoc.*, 96:331–338, 2001.

L. G. Telser. Iterative estimation of a set of linear regression equations. *J. Amer. Statist. Assoc.*, 59: 845–862, 1964.

N. Wermuth. Linear recursive equations, covariance selection, and path analysis. *J. Amer. Statist. Assoc.*, 75:963–972, 1980.

N. Wermuth and D. R. Cox. Joint response graphs and separation induced by triangular systems. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 66:687–717, 2004.

J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. Wiley, Chichester, 1990.

A. Wille, P. Zimmermann, E. Vranova, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelic, P. von Rohr, L. Thiele, E. Zitzler, W. Gruissem, and P. Bühlmann. Sparse graphical Gaussian modeling of the isoprenoid gene network in *arabidopsis thaliana*. *Genome Biology*, 5(11):R92, 2004.

S. Wright. Correlation and causation. *Journal of Agricultural Research*, 20:557–585, 1921.

S. Wright. The method of path coefficients. *Ann. Math. Statist.*, 5:161–215, 1934.

J. Zhang and P. Spirtes. A characterization of Markov equivalence classes for ancestral graphical models. Technical Report 168, Dept. of Philosophy, Carnegie-Mellon University, 2005.

Journal of Machine Learning Research 10 (2009) 2349-2374 Submitted 8/08; Revised 7/09; Published 10/09

Estimating Labels from Label Proportions*

Novi Quadrianto NOVI.QUAD@GMAIL.COM
Statistical Machine Learning
National ICT Australia
Locked Bag 8001
Canberra ACT 2601, Australia

Alex J. Smola ALEX@SMOLA.ORG
Yahoo! Research
4401 Great America Parkway
Santa Clara CA 95054, USA

Tibério S. Caetano TIBERIO.CAETANO@GMAIL.COM
Statistical Machine Learning
National ICT Australia
Locked Bag 8001
Canberra ACT 2601, Australia

Quoc V. Le QUOCLE@STANFORD.EDU
AI Lab
Department of Computer Science Department
Stanford University
Stanford CA 94305, USA

Editor: Tommi Jaakkola

Abstract

Consider the following problem: given sets of unlabeled observations, each set with known label proportions, predict the labels of another set of observations, possibly with known label proportions. This problem occurs in areas like e-commerce, politics, spam filtering and improper content detection. We present consistent estimators which can reconstruct the correct labels with high probability in a uniform convergence sense. Experiments show that our method works well in practice.

Keywords: unsupervised learning, Gaussian processes, classification and prediction, probabilistic models, missing variables

1. Introduction

Different types of learning problems assume different problem settings. In *supervised* learning, we are given sets of labeled instances. Another learning type called *unsupervised* learning focuses on the setting where unlabeled instances are given. Recently, it has been realized that unlabeled instances when used in conjunction with a small amount of labeled instances can deliver considerable learning performance improvement in comparison to using labeled instances alone. This leads to a *semi-supervised* learning setting.

∗. A short version of this paper appeared in Quadrianto et al. (2008).

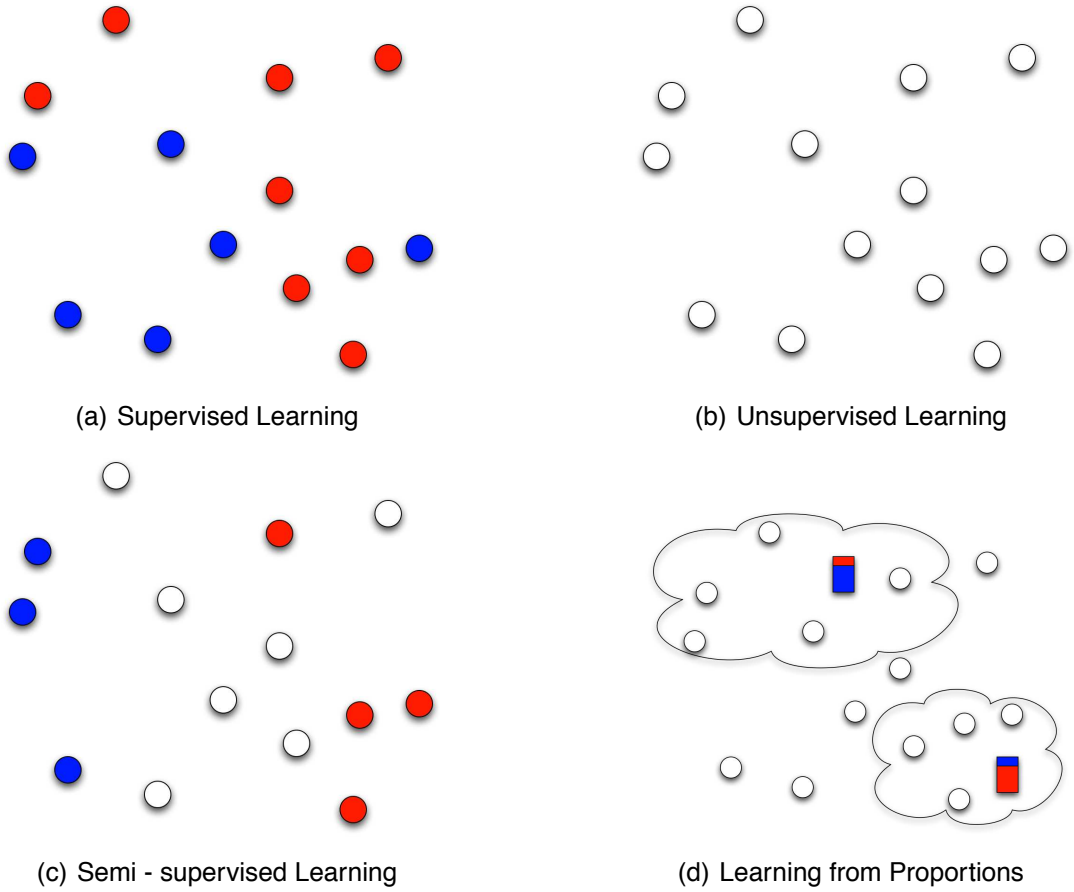


Figure 1: Different types of learning problems (colors encode class labels). 1(a) - `supervised learning`: only labeled instances are given; 1(b) - `unsupervised learning`: only unlabeled instances are given; 1(c) - `semi-supervised learning`: both labeled and unlabeled instances are given; 1(d): `learning from proportions`: at least as many data aggregates (groups of data with their associated class label proportions) as there are number of classes are given.

We are interested in a learning setting where groups of unlabeled instances are given. The number of group is at least as many as number of classes. Each group is equipped with information on class label *proportions*. We called this informative group as aggregate (see Figure 1 for an illustration). This type of learning problem appears in areas like e-commerce, politics, spam filtering and improper content detection, as we illustrate below.

Assume that an internet services company wants to increase its profit in sales. Obviously sending out discount coupons will increase sales, but sending coupons to customers who would have purchased the goods anyway decreases the margins. Alternatively, failing to send coupons to customers who would only buy in case of a discount reduces overall sales. We would like to identify the class of would-be customers who are most likely to change their purchase decision when receiving a coupon. The problem is that there is no direct access to a sample of would-be customers.

Typically only a sample of people who buy regardless of coupons (those who bought when there was no discount) and a mixed sample (those who bought when there was discount) are available. The mixing *proportions* can be reliably estimated using random assignment to control and treatment groups. How can we use this information to determine the would-be customers?

Politicians face the same problem. They can rely on a set of always-favorable voters who will favor them regardless, plus a set of swing voters who will make their decision dependent on what the candidates offer. Since the candidate's resources (finance, ability to make election promises, campaign time) are limited, it is desirable for them to focus their attention on that part of the demographic where they can achieve the largest gains. Previous elections can directly reveal the profile of those who favor regardless, that is those who voted in favor where low campaign resources were committed. Those who voted in favor where substantial resources were committed can be either swing voters or always-favorable. So in a typical scenario there is no separate sample of swing voters.

Likewise, consider the problem of spam filtering. Data sets of spam are likely to contain almost pure spam (this is achieved e.g. by listing e-mails as spam bait), while user's inboxes typically contain a mix of spam and non-spam. We would like to use the inbox data to improve estimation of spam. In many cases it is possible to estimate the *proportions* of spam and non-spam in a user's inbox much more cheaply than the actual labels. We would like to use this information to categorize e-mails into spam and non-spam.

Similarly, consider the problem of filtering images with "improper content". Data sets of such images are readily accessible thanks to user feedback, and it is reasonable to assume that this labeling is highly reliable. However the rest of images on the web (those not labeled) is a far larger data set, albeit without labels (after all, this is what we would like to estimate the labels for). That said, it is considerably cheaper to obtain a good estimate of the *proportions* of proper and improper content in addition to having one data set of images being of likely improper content. We would like to obtain a classifier based on this information.

2. Problem Definition

In this paper, we present a method that makes use of the knowledge of label proportions *directly*. As motivated by the above examples, our method would be practically useful in many domains such as identifying potential customers, potential voters, spam e-mails and improper images. We also prove bounds indicating that the estimates obtained are close to those from a fully labeled scenario.

Before defining the problem, we emphasize that the formal setting is more general than the above examples might suggest. More specifically, we may not require *any* label to be known, only their proportions within each of the involved data sets. Also the general problem is not restricted to the binary case but instead can deal with large numbers of classes. Finally, it is possible to apply our method to problems where the *test label proportions* are unknown, too. This simple modification allows us to use this technique whenever covariate shift via label bias is present.

Formally, in a learning from proportions setting, we are given n sets of observations $X_i = \{x_1^i, \ldots, x_{m_i}^i\}$ of respective sample sizes m_i (calibration set) $i = 1, \ldots, n$ as well as a set $X = \{x_1, \ldots, x_m\}$ (test set). Moreover, we are given the fractions π_{iy} of labels $y \in \mathcal{Y}$ ($|\mathcal{Y}| \leq n$) contained in each set X_i. These fractions form a full (column) rank mixing matrix, $\pi \in \mathbb{R}^{n \times |\mathcal{Y}|}$ with the constraint that each row sums up to 1 and all entries are nonnegative. The marginal probability $p(y)$ of the test set X may or may not be known. Note that the label dictionaries $\mathcal{Y}_i$ do not need to be the same

across all sets i (define $\mathcal{Y} := \cup_i \mathcal{Y}_i$) and we also allow for $\pi_{iy} = 0$ if needed. It is our goal to design algorithms which are able to obtain conditional class probability estimates $p(y|x)$ solely based on this information.

As an illustration, take the spam filtering example. We have $X_1 =$ "mail in spam box" (only spam) and $X_2 =$ "mail in inbox" (spam mixed with non-spam). Also suppose that we may know the proportion of spam vs non-spam in our inbox is $1:9$. That means, we know: $\pi_{1,\text{spam}} = 1.0, \pi_{1,\text{non-spam}} = 0, \pi_{2,\text{spam}} = 0.1$ and $\pi_{2,\text{non-spam}} = 0.9$. The test set X then may be X_2 itself, for example. Thus, the marginal probability of the test set will simply be: $p(y = \text{spam}) = 0.1, p(y = \text{non}-\text{spam}) = 0.9$. The goal is to find $p(\text{spam}|\text{mail})$ in X. Note that, in general, our setting is different and more difficult than that of transduction. The latter requires at least some labeled instances of *all classes* are given. In the spam filtering example, we have no pure non-spam instances.

Key to our proposed solution is a conditional independence assumption, $x \perp\!\!\!\perp i \,|\, y$. In other words, we assume that the *conditional* distribution of x is independent of the index i, as long as we know the label y. This is a crucial assumption: after all, we want the distributions within each class to be independent of which aggregate they can be found in. If this were not the case it would be impossible to infer about the distribution on the test set from the (biased) distributions over the aggregates.

3. Mean Operators

Our idea relies on uniform convergence properties of the expectation operator and of corresponding risk functionals (Altun and Smola, 2006; Dudík and Schapire, 2006). In doing so, we are able to design estimators with the same performance guarantees in terms of uniform convergence as those with full access to the label information.

At the heart of our reasoning lies the fact that many estimators rely on data by solving a convex optimization problem. We begin our exposition by discussing how this strategy can be employed in the context of exponential families. Subsequently we state convergence guarantees and we discuss how our method can be extended to other estimates such as Csiszar and Bregman divergences and other function spaces.

3.1 Exponential Families

Denote by $\mathcal{X}$ the space of observations and let $\mathcal{Y}$ be the space of labels. Moreover, let $\phi(x,y) : \mathcal{X} \times \mathcal{Y} \to \mathcal{H}$ be a feature map into a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ with kernel $k((x,y),(x',y'))$. In this case we may state conditional exponential models via

$$p(y|x,\theta) = \exp\left(\langle \phi(x,y),\theta\rangle - g(\theta|x)\right) \text{ with } g(\theta|x) = \log \sum_{y\in\mathcal{Y}} \exp\langle \phi(x,y),\theta\rangle,$$

where the normalization g is called the log-partition function, often referred to as the cumulant generating function. Note that while in general there is no need for $\mathcal{Y}$ to be discrete, we make this simplifying assumption in order to be able to reconstruct the class probabilities efficiently. For $\{(x_i,y_i)\}$ drawn iid from a distribution $p(x,y)$ on $\mathcal{X}\times\mathcal{Y}$ the conditional log-likelihood is given by

$$\log p(Y|X,\theta) = \sum_{i=1}^{m} \left[\langle \phi(x_i,y_i),\theta\rangle - g(\theta|x_i)\right] = m\,\langle \mu_{XY},\theta\rangle - \sum_{i=1}^{m} g(\theta|x_i),$$

where the empirical mean in feature space μ_{XY} is defined as in Table 2. In order to avoid overfitting one commonly maximizes the log-likelihood penalized by a prior $p(\theta)$. This means that we need to solve the following optimization problem

$$\theta^* := \operatorname*{argmin}_\theta \left[-\log\{p(Y|X,\theta)p(\theta)\}\right]. \tag{1}$$

For instance, for a Gaussian prior on θ, i.e. for

$$-\log p(\theta) = \lambda \|\theta\|^2 + \text{const.},$$

we have

$$\theta^* = \operatorname*{argmin}_\theta \left[\sum_{i=1}^m g(\theta|x_i) - m\langle \mu_{XY}, \theta\rangle + \lambda\|\theta\|^2\right]. \tag{2}$$

The problem is that in our setting we do not know the labels y_i, so the sufficient statistics μ_{XY} cannot be computed exactly. Note, though that the only place where the labels enter the estimation process is via the mean μ_{XY}. Our strategy is to exploit the fact that this quantity, however, is statistically well behaved and converges under relatively mild technical conditions at rate $O(m^{-\frac{1}{2}})$ to its expected value

$$\mu_{xy} := \mathbf{E}_{(x,y)\sim p(x,y)}[\phi(x,y)],$$

as will be shown in Theorem 3. Our goal therefore will be to estimate μ_{xy} and use it as a proxy for μ_{XY}, and only then solve (2) with the estimated $\hat{\mu}_{XY}$ instead of μ_{XY}. We will discuss explicit convergence guarantees in Section 5 after describing how to compute the mean operator in detail.

3.2 Estimating the Mean Operator

In order to obtain θ^* we would need μ_{XY}, which is impossible to compute exactly, since we do not have Y. However, we know that μ_{XY} converges to μ_{xy}. Hence, if we are able to approximate μ_{xy} then this, in turn, will be a good estimate for μ_{XY}.

Our quest is therefore as follows: express μ_{xy} as a linear combination over expectations with respect to the distributions on the data sets $X_1,\ldots,X_n$ (where $n \geq |\mathcal{Y}|$). Secondly, show that the expectations of the distributions having generated the sets X_i ($\mu_x^{\text{set}}[i,y']$, see Table 2), can be approximated by empirical means ($\mu_X^{\text{set}}[i,y']$, see Table 2). Finally, we need to combine both steps to provide guarantees for μ_{XY}.

It will turn out that in certain cases some of the algebra can be sidestepped, in particular whenever we may be able to identify several sets with each other (e.g. the test set X is one of the calibration data sets X_i) or whenever $\phi(x,y)$ factorizes into $\psi(x)\otimes\varphi(y)$. We will discuss these simplifications in Section 4.

3.2.1 MEAN OPERATOR

Since μ_{xy} is a linear operator mapping $p(x,y)$ into a Hilbert Space we may expand μ_{xy} via

$$\mu_{xy} = \mathbf{E}_{(x,y)\sim p(x,y)}[\phi(x,y)] = \sum_{y\in\mathcal{Y}} p(y)\mathbf{E}_{x\sim p(x|y)}[\phi(x,y)] = \sum_{y\in\mathcal{Y}} p(y)\mu_x^{\text{class}}[y,y],$$

X_i	i^{th} set of observations: $X_i = \{x_1^i, \ldots, x_{m_i}^i\}$
m_i	number of observations in X_i
X	test set of observations: $X = \{x_1, \ldots, x_m\}$
Y	test set of labels: $Y = \{y_1, \ldots, y_m\}$
m	number of observations in the test set X
π_{iy}	proportion of label y in set i
$\phi(x,y)$	map from (x,y) to a Hilbert Space

Table 1: Notations used in the paper.

Expectations with respect to the model:

$$\begin{aligned}
\mu_{xy} &:= \mathbf{E}_{(x,y)\sim p(x,y)}[\phi(x,y)] \\
\mu_x^{\text{class}}[y,y'] &:= \mathbf{E}_{(x)\sim p(x|y)}[\phi(x,y')] \\
\mu_x^{\text{set}}[i,y'] &:= \mathbf{E}_{(x)\sim p(x|i)}[\phi(x,y')] \\
\mu_x^{\text{class}}[y] &:= \mathbf{E}_{(x)\sim p(x|y)}[\psi(x)] \\
\mu_x^{\text{set}}[i] &:= \mathbf{E}_{(x)\sim p(x|i)}[\psi(x)]
\end{aligned}$$

Expectations with respect to data:

$$\begin{aligned}
\mu_{XY} &:= \tfrac{1}{m}\textstyle\sum_{i=1}^m \phi(x_i,y_i) \\
\text{(1a)}\quad \mu_X^{\text{set}}[i,y'] &:= \tfrac{1}{m_i}\textstyle\sum_{x\in X_i} \phi(x,y') \ \textbf{(known)} \\
\text{(1b)}\quad \mu_X^{\text{set}}[i] &:= \tfrac{1}{m_i}\textstyle\sum_{x\in X_i} \psi(x) \ \textbf{(known)}
\end{aligned}$$

Estimates:

$$\begin{aligned}
\text{(2)}\quad \hat{\mu}_x^{\text{class}} &= (\pi^\top\pi)^{-1}\pi^\top\mu_X^{\text{set}} \\
\text{(3a)}\quad \hat{\mu}_{XY} &= \textstyle\sum_{y\in\mathcal{Y}} p(y)\hat{\mu}_x^{\text{class}}[y,y] \\
\text{(3b)}\quad \hat{\mu}_{XY} &= \textstyle\sum_{y\in\mathcal{Y}} p(y)\varphi(y)\otimes\hat{\mu}_x^{\text{class}}[y] \\
\text{(4)}\quad \hat{\theta}^* &\text{ solution of (2) for } \mu_{XY} = \hat{\mu}_{XY}.
\end{aligned}$$

Table 2: Major quantities of interest in the paper. Numbers on the left represent the order in which the corresponding quantity is computed in the algorithm (letters denote the variant of the algorithm: 'a' for general feature map $\phi(x,y)$ and 'b' for factorizing feature map $\phi(x,y) = \psi(x)\otimes\varphi(y)$). Lowercase subscripts refer to model expectations, uppercase subscripts are sample averages.

where the shorthand $\mu_x^{\text{class}}[y,y]$ is defined in Table 2. This means that if we were able to compute $\mu_x^{\text{class}}[y,y]$ we would be able to "reassemble" μ_{xy} from its individual components. We now show that $\mu_x^{\text{class}}[y,y]$ can be estimated directly.

Our conditional independence assumption, $p(x|y,i) = p(x|y)$, yields the following:

$$p(x|i) = \sum_y p(x|y,i)p(y|i) = \sum_y p(x|y)\pi_{iy}. \tag{3}$$

In the above equation, we form a mixing matrix π with the element $\pi_{iy} = p(y|i)$. This allows us to define the following means

$$\mu_x^{\text{set}}[i,y'] := \mathbf{E}_{x\sim p(x|i)}[\phi(x,y')] \overset{(3)}{=} \sum_y \pi_{iy}\mu_x^{\text{class}}[y,y'].$$

Algorithm 1

Input data sets $X, \{X_i\}$, probabilities π_{iy} and $p(y)$
for $i = 1$ **to** n **and** $y' \in \mathcal{Y}$ **do**
  Compute empirical means $\mu_X^{\text{set}}[i, y']$
end for
Compute $\hat{\mu}_x^{\text{class}} = (\pi^\top \pi)^{-1} \pi^\top \mu_X^{\text{set}}$
Compute $\hat{\mu}_{XY} = \sum_{y \in \mathcal{Y}} p(y) \hat{\mu}_x^{\text{class}}[y, y]$
Solve the minimization problem

$$\hat{\theta}^* = \operatorname*{argmin}_{\theta} \left[\sum_{i=1}^{m} g(\theta | x_i) - m \langle \hat{\mu}_{XY}, \theta \rangle + \lambda \|\theta\|^2 \right]$$

Return $\hat{\theta}^*$.

Note that in order to compute $\mu_x^{\text{set}}[i, y']$ we do *not* need any label information with respect to $p(x|i)$. It is simply the expectation of $\phi(\cdot, y')$ on the distribution of bag i. However, since we have at least $|\mathcal{Y}|$ of those equations and we assumed that π has full column rank, they allow us to solve a linear system of equations and compute $\mu_x^{\text{class}}[y, y]$ from $\mu_x^{\text{set}}[i, y']$ for all i. In shorthand we may use

$$\mu_x^{\text{set}} = \pi \mu_x^{\text{class}} \text{ and hence } \mu_x^{\text{class}} = (\pi^\top \pi)^{-1} \pi^\top \mu_x^{\text{set}} \tag{4}$$

to compute $\mu_x^{\text{class}}[y, y]$ for all $y \in \mathcal{Y}$. With some slight abuse of notation we have μ_x^{class} and μ_x^{set} represent the *matrices* of terms $\mu_x^{\text{class}}[y, y']$ and $\mu_x^{\text{set}}[i, y']$ respectively. There will be as many matrices as the dimensions of $\phi(x, y)$, thus (4) has to be solved separately for each dimension of $\phi(x, y)$.

Obviously we cannot compute $\mu_x^{\text{set}}[i, y']$ explicitly, since we only have *samples* from $p(x|i)$. However the same convergence results governing the convergence of μ_{XY} to μ_{xy} also hold for the convergence of $\mu_X^{\text{set}}[i, y']$ to $\mu_x^{\text{set}}[i, y']$. Hence we may use the empirical average $\mu_X^{\text{set}}[i, y']$ as the estimate for $\mu_x^{\text{set}}[i, y']$ and from that find an estimate for μ_{XY}.

3.2.2 BIG PICTURE

Overall, our strategy is as follows: use empirical means on the bags X_i to approximate expectations with respect to the bag distribution. Use the latter to compute expectations with respect to a given label, and finally, use the means conditional on the label distribution to obtain μ_{xy} which is a good proxy for μ_{XY} (see Algorithm 1), i.e.

$$\mu_X^{\text{set}}[i, y'] \longrightarrow \mu_x^{\text{set}}[i, y'] \longrightarrow \mu_x^{\text{class}}[y, y'] \longrightarrow \mu_{xy} \longrightarrow \mu_{XY}.$$

For the first and last step in the chain we can invoke uniform convergence results. The remaining two steps in the chain follow from linear algebra. As we shall see, whenever there are considerably more bags than classes we can exploit the overdetermined system to our advantage to reduce the overall estimation error and use a rescaled version of (4).

4. Special Cases

In some cases the calculations described in Algorithm 1 can be carried out more efficiently. They arise whenever the matrix π has special structure or whenever the test set and one of the training

sets coincide. Moreover, we may encounter situations where the fractions of observations in the test set are unknown and we would like, nonetheless, to find a good proxy for μ_{XY}.

4.1 Minimal Number of Sets

Assuming that $|\mathcal{Y}| = n$ and that π has full rank it follows that $(\pi^\top\pi)^{-1}\pi^\top = \pi^{-1}$. Hence we can obtain the proxy for μ_{XY} more directly via $\mu_x^{\text{class}} = \pi^{-1}\mu_x^{\text{set}}$.

4.2 Testing on One of the Calibration Sets

Note that there is no need for requiring that the test set X be different from one of the calibration sets (vide example in Problem Definition). In particular, when $X = X_i$ the uncertainty in the estimate of μ_{XY} can be greatly reduced provided that the estimate of μ_{XY} as given in (4) contains a large fraction of the mean of at least one of the classes. We will discuss this situation in more detail when it comes to binary classification since there the advantages will be most obvious.

4.3 Special Feature Map

Whenever the feature map $\phi(x,y)$ factorizes into $\psi(x) \otimes \varphi(y)$ we can simplify calculation of the means considerably. More specifically, instead of estimating $O(|\mathcal{Y}| \cdot n)$ parameters we only require calculation of $O(n)$ terms. The reason for this is that we may pull the dependency on y out of the expectations. Defining $\mu_x^{\text{class}}[y], \mu_x^{\text{set}}[i]$, and $\mu_X^{\text{set}}[i]$ as in Table 2 allows us to simplify

$$\hat{\mu}_{XY} = \sum_{y \in \mathcal{Y}} p(y)\varphi(y) \otimes \hat{\mu}_x^{\text{class}}[y] \text{ where } \hat{\mu}_x^{\text{class}} = (\pi^\top\pi)^{-1}\pi^\top\mu_X^{\text{set}}. \tag{5}$$

Here the last equation is understood to apply to the vector of means $\mu_x := (\mu[1], \ldots, \mu[n])$ and μ_X accordingly. A significant advantage of (5) is that we only need to perform $O(n)$ averaging operations rather than $O(n \cdot |\mathcal{Y}|)$. Obviously the cost of computing $(\pi^\top\pi)^{-1}\pi^\top$ remains unchanged but the latter is negligible compared to the operations in Hilbert Space. Note that $\psi(x) \in \mathbb{R}^D$ denotes an arbitrary feature representation of the inputs, which in many cases can be defined implicitly via a kernel function. As the joint feature map $\phi(x,y)$ factorizes into $\psi(x) \otimes \varphi(y)$, we can write the inner product in the joint representation as $\langle \phi(x,y), \phi(x',y') \rangle = \langle \psi(x), \psi(x') \rangle \langle \varphi(y), \varphi(y') \rangle = k(x,x')k(y,y')$. In general, the kernel function on inputs and labels can be different. Specifically, for a label diagonal kernel $k(y,y') = \delta(y,y')$, the standard winner-takes-all multiclass classification is recovered (Tsochantaridis et al., 2005). With this setting, the input feature $\psi(x)$ can be defined implicitly via a kernel function by invoking the Representer Theorem (Schölkopf and Smola, 2002).

4.4 Binary Classification

One may show (Hofmann et al., 2006) that the feature map $\phi(x,y)$ takes on a particularly appealing form of $\phi(x,y) = y\psi(x)$ where $y \in \{\pm 1\}$. This follows since we can always re-calibrate $\langle \phi(x,y), \theta \rangle$ by an offset independent of y such that $\phi(x,1) + \phi(x,-1) = 0$.

If we moreover assume that X_1 only contains class 1 and $X_2 = X$ contains a mixture of classes with labels 1 and -1 with proportions $p(1) =: \rho$ and $p(-1) = 1 - \rho$ respectively, we obtain the mixing matrix

$$\pi = \begin{bmatrix} 1 & 0 \\ \rho & 1-\rho \end{bmatrix} \Rightarrow \pi^{-1} = \begin{bmatrix} 1 & 0 \\ \frac{-\rho}{1-\rho} & \frac{1}{1-\rho} \end{bmatrix}.$$

Plugging this into (5) yields

$$\begin{aligned}\hat{\mu}_{XY} &= \rho\mu_X^{\text{set}}[1] - (1-\rho)\left[\tfrac{-\rho}{1-\rho}\mu_X^{\text{set}}[1] + \tfrac{1}{1-\rho}\mu_X^{\text{set}}[2]\right] \\ &= 2\rho\mu_X^{\text{set}}[1] - \mu_X^{\text{set}}[2]. \end{aligned} \tag{6}$$

Consequently, taking a simple weighted difference between the averages on two sets, e.g. one set containing spam whereas the other one containing an unlabeled mix of spam and non-spam, allows one to obtain the sufficient statistics needed for estimation.

4.5 Overdetermined Systems

Assume that we have significantly more bags n than class labels $|\mathcal{Y}|$, possibly with varying numbers of observations m_i per bag. In this case it would make sense to find a weighting of the bags such that those which are largest and most relevant for the test set are given the highest degree of importance. Instead of stating the problem as one of solving a linear system we now restate it as one of solving an approximation problem. To simplify notation we assume that the feature map factorizes, i.e. that $\phi(x,y) = \psi(x) \otimes \varphi(y)$. A weighted linear combination of the squared discrepancy between the class means and the set means is given by

$$\underset{\mu_x^{\text{class}}}{\text{minimize}} \sum_{i=1}^{n} w_i \left\| \mu_X^{\text{set}}[i] - \sum_{y\in\mathcal{Y}} \pi_{iy}\mu_x^{\text{class}}[y] \right\|^2, \tag{7}$$

where w_i are some previously chosen weights which reflect the importance of each bag. Typically we might choose $w_i = O(m_i^{-\frac{1}{2}})$ to reflect the fact that convergence between empirical means and expectations scales with $O(m^{-\frac{1}{2}})$. Before we discuss specific methods for choosing a weighting, let us review the statistical properties of the estimator.

Remark 1 (Underdetermined Systems) *Similarly, when we have less bags n than class labels $|\mathcal{Y}|$, we can state the problem as one of solving a regularized least squares problem as follows*

$$\underset{\mu_x^{\text{class}}}{\text{minimize}} \sum_{i=1}^{n} \left\| \mu_X^{\text{set}}[i] - \sum_{y\in\mathcal{Y}} \pi_{iy}\mu_x^{\text{class}}[y] \right\|^2 + \lambda\Omega(\mu_x^{\text{class}}[y]\,\forall y \in \mathcal{Y}).$$

For example, we can let $\Omega(\mu_x^{\text{class}}[y]\,\forall y \in \mathcal{Y}) = \sum_{y\in\mathcal{Y}} \left\|\mu_x^{\text{class}}[y] - \mu_x^{\text{class}}[y+1]\right\|^2$. This makes sense whenever different labels have related means $\mu_x^{\text{class}}[y]$.

5. Convergence Bounds

The obvious question is how well $\hat{\mu}_{XY}$ manages to approximate μ_{XY} and secondly, how badly any error in estimating μ_{XY} would affect the overall quality of the solution. We approach this problem as follows: first we state the uniform convergence properties of μ_{XY} and similar empirical operators relative to μ_{xy}. Secondly, we apply those bounds to the cases discussed above, and thirdly, we show that the approximate minimizer of the log-posterior has a bounded deviation from what we would have obtained by knowing μ_{XY} exactly. Much of the reasoning follows the ideas of Altun and Smola (2006).

5.1 Uniform Convergence for Mean Operators

An important tool in studying uniform convergence properties of random variables are Rademacher averages (Ledoux and Talagrand, 1991; Mendelson, 2002). They are needed to state the key results in our context.

Definition 2 (Rademacher Averages) *Let $\mathcal{X}$ be a domain and p a distribution on $\mathcal{X}$ and assume that $X := \{x_1, \ldots, x_m\}$ is drawn iid from p. Moreover, let $\mathcal{F}$ be a class of functions $\mathcal{X} \to \mathbb{R}$. Furthermore denote by σ_i Rademacher random variables, i.e. $\{\pm 1\}$ valued with zero mean. The Rademacher average is*

$$R_m(\mathcal{F}, p) := \mathbf{E}_X \mathbf{E}_\sigma \left[\sup_{f \in \mathcal{F}} \left| \frac{1}{m} \sum_{i=1}^{m} \sigma_i f(x_i) \right| \right].$$

This quantity measures the flexibility of the function class $\mathcal{F}$ —in our case linear functions in $\phi(x,y)$. Altun and Smola (2006) state the following result:

Theorem 3 (Convergence of Empirical Means) *Denote by $\phi : \mathcal{X} \to \mathcal{B}$ a map into a Banach space $\mathcal{B}$, denote by $\mathcal{B}^*$ its dual space and let $\mathcal{F}$ the class of linear functions on $\mathcal{B}$ with bounded $\mathcal{B}^*$ norm by 1. Let $R > 0$ such that for all $f \in \mathcal{F}$ we have $|f(x)| \leq R$. Moreover, assume that X is an m-sample drawn from p on $\mathcal{X}$. For $\bar{\varepsilon} > 0$ we have that with probability at least $1 - \exp(-\bar{\varepsilon}^2 m / 2R^2)$ the following holds:*

$$\|\mu_X - \mu_x\|_{\mathcal{B}} \leq 2R_m(\mathcal{F}, p) + \bar{\varepsilon}.$$

For $k \geq 0$ we only have a failure probability of $1 - \exp(-\bar{\varepsilon}^2 m / R^2)$.

Theorem 4 (Bartlett and Mendelson 2002) *Whenever $\mathcal{B}$ is a Reproducing Kernel Hilbert Space with kernel $k(x,x')$ the Rademacher average can be bounded from above by $R_m(\mathcal{F}) \leq m^{-\frac{1}{2}} \left[\mathbf{E}_x[k(x,x)] \right]^{\frac{1}{2}}$.*

Our approximation error can be bounded as follows. From the triangle inequality we have:

$$\|\hat{\mu}_{XY} - \mu_{XY}\| \leq \|\hat{\mu}_{XY} - \mu_{xy}\| + \|\mu_{xy} - \mu_{XY}\|.$$

For the second term we may employ Theorem 3 directly. To bound the first term note that by linearity

$$\varepsilon := \hat{\mu}_{XY} - \mu_{xy} = \sum_y p(y) \left[(\pi^\top \pi)^{-1} \pi^\top \hat{\varepsilon} \right]_{y,y}, \tag{8}$$

where we define the "matrix" of coefficients

$$\hat{\varepsilon}\left[i, y'\right] := \mu_x^{\text{set}}[i, y'] - \mu_X^{\text{set}}[i, y']. \tag{9}$$

In the more general case of overdetermined systems we have

$$\varepsilon = \sum_y p(y) \left[(\pi^\top W \pi)^{-1} \pi^\top W \hat{\varepsilon} \right]_{y,y}.$$

Now note that all $\hat{\mathfrak{e}}[i,y']$ also satisfy the conditions of Theorem 3 since the sets X_i are drawn iid from the distributions $p(x|i)$ respectively. We may bound each term individually in this fashion and subsequently apply the union bound to ensure that all $n \cdot |\mathcal{Y}|$ components satisfy the constraints. Hence each of the terms needs to satisfy the constraint with probability $1-\delta/(n|\mathcal{Y}|)$ to obtain an overall bound with probability $1-\delta$. To obtain bounds we would need to bound the linear operator mapping $\hat{\mathfrak{e}}$ into $\mathfrak{e}$.

Note that this statement can be improved since all errors $\hat{\mathfrak{e}}[i,y']$ and $\hat{\mathfrak{e}}[j,y']$ for $i \neq j$ are independent of each other simply by the fact that each bag X_i was sampled independently from the other. We will discuss this in the context of choosing a practically useful value of W below.

5.2 Special Cases

A closed form solution in the general case is not particularly useful since it depends heavily on the kernel k, the mixing proportions π and the class probabilities on the test set. However, for a number of special cases it is possible to provide more detailed explicit analysis: firstly the situation where $\phi(x,y) = \psi(x) \otimes \varphi(y)$ and secondly, the binary classification setting where $\phi(x,y) = y\psi(x)$ and $X_2 = X$, where much tighter bounds are available.

5.3 Special Feature Map with Full Rank

Here we only need to deal with n rather than with $n \times |\mathcal{Y}|$ empirical estimates, i.e. $\mu_X^{\text{set}}[i]$ vs. $\mu_X^{\text{set}}[i,y']$. Hence (8) and (9) specialize to

$$\begin{aligned}\mathfrak{e} &= \sum_y p(y) \sum_{i=1}^{n} \varphi(y) \otimes \left[(\pi^\top \pi)^{-1}\pi^\top\right]_{yi} \hat{\mathfrak{e}}[i] \\ \hat{\mathfrak{e}}[i] &:= \mu_x^{\text{set}}[i] - \mu_X^{\text{set}}[i].\end{aligned}$$

Assume that with high probability each $\hat{\mathfrak{e}}[i]$ satisfies $\|\hat{\mathfrak{e}}[i]\| \leq c_i$ (we will deal with the explicit constants c_i later). Moreover, assume for simplicity that $|\mathcal{Y}| = n$ and that π has full rank (otherwise we need to follow through on our expansion using $(\pi^\top\pi)^{-1}\pi^\top$ instead of π^{-1}). This implies that

$$\begin{aligned}\|\mathfrak{e}\|^2 &= \sum_{i,j} \langle \hat{\mathfrak{e}}[i], \hat{\mathfrak{e}}[j] \rangle \times \sum_{y,y'} p(y)p(y')k(y,y') \left[\pi^{-1}\right]_{yi} \left[\pi^{-1}\right]_{y'j} \\ &\leq \sum_{i,j} c_i c_j \left| \left[\pi^{-1}\right]^\top K^{y,p} \pi^{-1} \right|_{ij}, \end{aligned} \tag{10}$$

where $K^{y,p}_{y,y'} = k(y,y')p(y)p(y')$. Combining several bounds we have the following theorem:

Theorem 5 *Assume that we have n sets of observations X_i of size m_i, each of which drawn from distributions with probabilities π_{iy} of observing data with label y. Moreover, assume that $k((x,y),(x',y')) = k(x,x')k(y,y') \geq 0$ where $k(x,x) \leq 1$ and $k(y,y) \leq 1$. Finally, assume that $m = |X|$. In this case the mean operator μ_{XY} can be estimated by $\hat{\mu}_{XY}$ with probability at least $1-\delta$ with precision*

$$\|\mu_{XY} - \hat{\mu}_{XY}\| \leq \left[2 + \sqrt{\log((n+1)/\delta)}\right] \times \left[m^{-\frac{1}{2}} + \left[\sum_{i,j} m_i^{-\frac{1}{2}} m_j^{-\frac{1}{2}} \left| \left[\pi^{-1}\right]^\top K^{y,p} \pi^{-1} \right|_{ij}\right]^{\frac{1}{2}}\right].$$

Proof We begin our argument by noting that both for $\phi(x,y)$ and for $\psi(x)$ the corresponding Rademacher averages R_m for functions of RKHS norm bounded by 1 is bounded by $m^{-\frac{1}{2}}$. This is a consequence of all kernels being bounded by 1 in Theorem 4 and $k \geq 0$.

Next note that in Theorem 3 we may set $R = 1$, since for $\|f\| \leq 1$ and $k((x,y),(x,y)) \leq 1$ and $k(x,x) \leq 1$ it follows from the Cauchy Schwartz inequality that $|f(x)| \leq 1$. Solving $\delta \leq \exp -m\varepsilon^2$ for ε yields $\varepsilon \leq m^{-\frac{1}{2}} \left[2 + \sqrt{\log(1/\delta)}\right]$.

Finally, note that we have $n+1$ deviations which we need to bound: one between μ_{XY} and μ_{xy}, and n for each of the $\varepsilon[i]$ respectively. Dividing the failure probability δ into $n+1$ cases yields bounds of the form $m^{-\frac{1}{2}} \left[2 + \sqrt{\log((n+1)/\delta)}\right]$ and $m_i^{-\frac{1}{2}} \left[2 + \sqrt{\log((n+1)/\delta)}\right]$ respectively. Plugging all error terms into (10) and summing over terms yields the claim and substituting this back into the triangle inequality proves the claim. ■

5.4 Binary Classification

Next we consider the special case of binary classification where $X_2 = X$. Using (6) we see that the corresponding estimator is given by

$$\hat{\mu}_{XY} = 2\rho\mu_X^{\text{set}}[1] - \mu_X^{\text{set}}[2].$$

Since $\hat{\mu}_{XY}$ shares a significant fraction of terms with μ_{XY} we are able to obtain tighter bounds as follows:

Theorem 6 *With probability* $1-\delta$ *(for* $1 > \delta > 0$*) the following bound holds:*

$$\|\hat{\mu}_{XY} - \mu_{XY}\| \leq 2\rho \left[2 + \sqrt{\log(2/\delta)}\right] \left[m_1^{-\frac{1}{2}} + m_+^{-\frac{1}{2}}\right],$$

where m_+ *is the number of observations with* $y = 1$ *in* X_2.

Proof Denote by $\mu[X_+]$ and $\mu[X_-]$ the averages over the subsets of X_2 with positive and negative labels respectively. By construction we have that

$$\begin{aligned}\mu_{XY} &= \rho\mu[X_+] - (1-\rho)\mu[X_-] \\ \hat{\mu}_{XY} &= 2\rho\mu_X^{\text{set}}[1] - \rho\mu[X_+] - (1-\rho)\mu[X_-].\end{aligned}$$

Taking the difference yields $2\rho\left[\mu_X^{\text{set}}[1] - \mu[X_+]\right]$. To prove the claim note that we may use Theorem 3 both for $\left\|\mu_X^{\text{set}}[1] - \mathbf{E}_{x\sim p(x|y=1)}[\psi(x)]\right\|$ and for $\left\|\mu[X_+] - \mathbf{E}_{x\sim p(x|y=1)}[\psi(x)]\right\|$. Taking the union bound and summing over terms proves the claim. ■

The bounds we provided show that $\hat{\mu}_{XY}$ converges at the same rate to μ_{xy} as μ_{XY} does, assuming that the sizes of the sets X_i increase at the same rate as X.

5.5 Overdetermined Systems

Given the optimal value of weighting W, the class mean can be reconstructed as a solution of a weighted least square problem in (7) and this minimizer is given by

$$\hat{\mu}_x^{\text{class}} = (\pi^\top W \pi)^{-1} \pi^\top W \mu_X^{\text{set}} \text{ where } W = \text{diag}(w_1, \ldots, w_n) \text{ and } w_i > 0.$$

It is easy to see that whenever $n = |\mathcal{Y}|$ and π has full rank there is only one possible solution regardless of the choice of W. For overdetermined systems the choice of W may greatly affect the

quality of the solution and it is therefore desirable to choose a weighting which minimizes the error in estimating μ_{XY}.

In choosing a weighting, we may take advantage of the fact that the errors $\hat{\varepsilon}[i]$ are independent for all i. This follows from the fact that all bags are drawn independently of each other. Moreover, we know that $\mathbf{E}[\hat{\varepsilon}[i]] = 0$ for all i. Finally we make the assumption that $k(y,y') = \delta(y,y')$, that is, that the kernel in the labels is diagonal. In this situation our analysis is greatly simplified and we have:

$$\varepsilon = \sum_y \varphi(y) \otimes p(y)(\pi^\top W \pi)^{-1} \pi W \hat{\varepsilon}$$

$$\text{and hence } \mathbf{E}\left[\|\varepsilon\|^2\right] = \sum_{i=1}^n \sum_y \mathbf{E}\left[\|\hat{\varepsilon}[i]\|^2\right] W_{ii}^2 \left[\pi_i^\top (\pi^\top W \pi)^{-1}\right]_y^2 p^2(y).$$

Using the assumption that $\mathbf{E}\left[\|\hat{\varepsilon}[i]\|^2\right] = O(m_i^{-1})$ we may find a suitable scale of the weight vectors by minimizing

$$\sum_{i=1}^n \sum_y \frac{W_{ii}^2}{m_i} \left[\pi_i^\top (\pi^\top W \pi)^{-1}\right]_y^2 p^2(y) \tag{11}$$

with respect to the diagonal matrix W. Note that the optimal value of W depends *both* on the mixtures of the bags π_i *and* on the propensity of each class $p(y)$. That is, being able to well estimate a class which hardly occurs at all is of limited value.

5.6 Stability Bounds

To complete our reasoning we need to show that our bounds translate into guarantees in terms of the minimizer of the log-posterior. In other words, estimates using the correct mean μ_{XY} vs. its estimate $\hat{\mu}_{XY}$ do not differ by a significant amount. For this purpose we make use of Altun and Smola (2006, Lemma 17).

Lemma 7 *Denote by f a convex function on $\mathcal{H}$ and let $\mu, \hat{\mu} \in \mathcal{H}$. Moreover let $\lambda > 0$. Finally denote by $\theta^*, \in \mathcal{H}$ the minimizer of*

$$L(\theta, \mu) := f(\theta) - \langle \mu, \theta \rangle + \lambda \|\theta\|^2$$

with respect to θ and $\hat{\theta}^*$ the minimizer of $L(\hat{\theta}, \hat{\mu})$ respectively. In this case the following inequality holds:

$$\left\|\theta^* - \hat{\theta}^*\right\| \leq \lambda^{-1} \left\|\mu - \hat{\mu}\right\|. \tag{12}$$

This means that a good estimate for μ immediately translates into a good estimate for the minimizer of the approximate log-posterior. This leads to the following bound on the risk minimizer.

Corollary 8 *The deviation between θ^*, as defined in (1) and $\hat{\theta}^*$, the minimizer of the approximate log-posterior using $\hat{\mu}_{XY}$ rather than μ_{XY}, is bounded by $O(m^{-\frac{1}{2}} + \sum_i m_i^{-\frac{1}{2}})$.*

Finally, we may use Altun and Smola (2006, Theorem 16) to obtain bounds on the quality of $\hat{\theta}^*$ when considering how well it minimizes the *true* negative log-posterior. Using the bound

$$L(\hat{\theta}^*, \mu) - L(\theta^*, \mu) \leq \left\|\hat{\theta}^* - \theta^*\right\| \left\|\hat{\mu} - \mu\right\|$$

yields the following bound for the log-posterior:

Corollary 9 *The minimizer $\hat{\theta}^*$ of the approximate log-posterior using $\hat{\mu}_{XY}$ rather than μ_{XY} incurs a penalty of at most $\lambda^{-1}\|\hat{\mu}_{XY}-\mu_{XY}\|^2$.*

5.7 Stability Bounds under Perturbation

Denote $1 \in \{1\}^{|\mathcal{Y}|}$ as the vector of all ones and $0 \in \{0\}^{|\mathcal{Y}|}$ as the vector of all zeros. Let Δ be the perturbation matrix such that the perturbed mixing matrix $\tilde{\pi}$ is related to the original mixing matrix π by $\tilde{\pi} = \pi + \Delta$. Note that the perturbed mixing matrix $\tilde{\pi}$ still needs to have non-negative entries and each row sums up to $1, \tilde{\pi}1 = 1$. The stochasticity constraint on the perturbed mixing matrix imposes special structure on the perturbation matrix, i.e. each row of perturbation matrix must sum up to 0, $\Delta 1 = 0$. Let $\hat{\theta}^*$ be the minimizer of (2) with mean $\hat{\mu}_{XY}$ approximated via mixing matrix π. Similarly, define $\tilde{\theta}^*$ for $\tilde{\mu}_{XY}$ with mixing matrix $\tilde{\pi}$. We would like to bound the distance $\left\|\hat{\theta}^* - \tilde{\theta}^*\right\|$ between the minimizers. Our perturbation bound relies on Lemma 7 and on the fact that we can bound the errors made in computing an (pseudo-) inverse of a matrix:

Lemma 10 (Stability of Inverses) *For any matrix norm $\|.\|$ and full rank matrices π and $\pi+\Delta$, the error between the inverses of π and $\pi+\Delta$ is bounded by*

$$\left\|\pi^{-1} - (\pi+\Delta)^{-1}\right\| \leq \left\|\pi^{-1}\right\| \left\|(\pi+\Delta)^{-1}\right\| \|\Delta\|.$$

Proof We use the following identity $\pi^{-1} - (\pi+\Delta)^{-1} = (\pi+\Delta)^{-1}\Delta\pi^{-1}$. The identity can be shown by left multiplying both sides of equation with $(\pi+\Delta)$. Finally, by submultiplicative property of a matrix norm, the inequality $\left\|\pi^{-1}\Delta(\pi+\Delta)^{-1}\right\| \leq \left\|\pi^{-1}\right\| \|\Delta\| \left\|(\pi+\Delta)^{-1}\right\|$ follows. ■

Theorem 11 (Stability of Pseudo-Inverses: Wedin 1973) *For any unitarily invariant matrix norm $\|.\|$ and full column rank matrices π and $\pi+\Delta$, the error between the pseudo-inverses of π and $\pi+\Delta$ is bounded by*

$$\left\|\pi^{\dagger} - (\pi+\Delta)^{\dagger}\right\| \leq \mu \left\|\pi^{\dagger}\right\|_{\sigma\infty} \left\|(\pi+\Delta)^{\dagger}\right\|_{\sigma\infty} \|\Delta\|,$$

where μ denotes a scalar constant depending on the matrix norm, $\|.\|_{\sigma\infty}$ denotes the spectral norm of a matrix, and the pseudo-inverse $\pi^{\dagger}$ defined as $\pi^{\dagger} := (\pi^{\top}\pi)^{-1}\pi^{\top}$.

Proof See Wedin (1973, Theorem 4.1) for a proof. ■

Remark 12 *For full rank matrices, the constant term μ in Theorem 11 is equal to unity regardless of the matrix norm considered (Wedin, 1973).*

First, we would like to bound the difference between $\hat{\mu}_{XY}$ and $\tilde{\mu}_{XY}$, i.e. $\varepsilon_p := \hat{\mu}_{XY} - \tilde{\mu}_{XY}$. For the special feature map with full rank, this translates to

$$\varepsilon_p = \sum_y p(y) \sum_{i=1}^{n} \varphi(y) \otimes \left[\pi^{-1} - \tilde{\pi}^{-1}\right]_{yi} \mu_X^{\text{set}}[i]$$

$$\|\varepsilon_p\|^2 = \sum_{i,j} \left\langle \mu_X^{\text{set}}[i], \mu_X^{\text{set}}[j] \right\rangle \times \left[(\pi^{-1} - \tilde{\pi}^{-1})^{\top} K^{y,p} (\pi^{-1} - \tilde{\pi}^{-1})\right]_{ij}.$$

Lemma 13 *Define $K^{y,p} := V_{y,p}^\top V_{y,p}$. With the spectral norm $\|.\|_{\sigma\infty}$ and a full rank mixing matrix π, the following bound holds:*

$$\|\hat{\mu}_{XY} - \tilde{\mu}_{XY}\|_{\sigma\infty} \leq \|V_{y,p}\|_{\sigma\infty} \left\|\pi^{-1}\right\|_{\sigma\infty} \|\Delta\|_{\sigma\infty} \left\|(\pi+\Delta)^{-1}\right\|_{\sigma\infty} \left[\sum_{i,j} \left\langle \mu_X^{set}[i], \mu_X^{set}[j] \right\rangle\right]^{\frac{1}{2}}. \quad (13)$$

Proof We first upper bound $\left[(\pi^{-1} - \tilde{\pi}^{-1})^\top K^{y,p} (\pi^{-1} - \tilde{\pi}^{-1})\right]_{ij}$ by $\left\|(\pi^{-1} - \tilde{\pi}^{-1})^\top K^{y,p} (\pi^{-1} - \tilde{\pi}^{-1})\right\|_{\sigma\infty}$. We factorize $K^{y,p}$ as $V_{y,p}^\top V_{y,p}$ since $K^{y,p}$ is a positive (semi-) definite matrix. The element $K^{y,p}_{y,y'} = k(y,y')p(y)p(y')$ is obtained by multiplying a kernel $k(y,y')$ with a rank-one kernel $k'(y,y') = p(y)p(y')$ where p is a positive function. This conformal transformation preserves the positive (semi-) definiteness of $K^{y,p}$ (Schölkopf and Smola, 2002). Thus, $\left\|(\pi^{-1} - \tilde{\pi}^{-1})^\top K^{y,p} (\pi^{-1} - \tilde{\pi}^{-1})\right\|_{\sigma\infty} \leq \left\|V_{y,p}(\pi^{-1} - \tilde{\pi}^{-1})\right\|_{\sigma\infty}^2 \leq$ $\left[\|V_{y,p}\|_{\sigma\infty} \left\|(\pi^{-1} - \tilde{\pi}^{-1})\right\|_{\sigma\infty}\right]^2 \leq \left[\|V_{y,p}\|_{\sigma\infty} \left\|\pi^{-1}\right\|_{\sigma\infty} \|\Delta\|_{\sigma\infty} \left\|(\pi+\Delta)^{-1}\right\|_{\sigma\infty}\right]^2$. The last inequality follows directly from Lemma 10. ■

Corollary 14 *Define $K^{y,p} := V_{y,p}^\top V_{y,p}$. With the spectral norm $\|.\|_{\sigma\infty}$ and a full column rank mixing matrix π, the following bound holds:*

$$\|\hat{\mu}_{XY} - \tilde{\mu}_{XY}\|_{\sigma\infty} \leq \sqrt{2}\,\|V_{y,p}\|_{\sigma\infty} \left\|\pi^{\dagger}\right\|_{\sigma\infty} \|\Delta\|_{\sigma\infty} \left\|(\pi+\Delta)^{\dagger}\right\|_{\sigma\infty} \left[\sum_{i,j} \left\langle \mu_X^{set}[i], \mu_X^{set}[j] \right\rangle\right]^{\frac{1}{2}}. \quad (14)$$

Proof Similar to Lemma 13 with the constant factor μ in Theorem 11 equals to $\sqrt{2}$ for a spectral norm. ■

Combining Lemma 13 for the full rank mixing matrix case (or Corollary 14 for the full column rank mixing matrix case) with Lemma 7, we are ready to state the stability bound under perturbation:

Lemma 15 (Stability Bound under Perturbation) *The distance ε_s between the two minimizers, $\hat{\theta}^*$ and $\tilde{\theta}^*$, is bounded by*

$$\varepsilon_s \leq \lambda^{-1} \|\hat{\mu}_{XY} - \tilde{\mu}_{XY}\|.$$

It is clear from (13) and (14) that the stability of our algorithm under perturbation will depend on the size of the perturbation and on the behavior of the (pseudo-) inverse of the perturbed mixing matrix. Note that by the triangle inequality, the distance in (12) can be decomposed as $\left\|\theta^* - \hat{\theta}^*\right\| \leq \left\|\theta^* - \tilde{\theta}^*\right\| + \left\|\tilde{\theta}^* - \hat{\theta}^*\right\|$ and the second term in RHS vanishes whenever the size of perturbation Δ is zero.

6. Extensions

We describe two types of extensions on our proposed estimator: function spaces and unknown label proportions on the test sets. We will discuss both of them in turn.

6.1 Function Spaces

Note that our analysis so far focused on a specific setting, namely maximum-a-posteriori analysis in exponential families. While this is a common and popular setting, the derivations are by no means restricted to this. We have the entire class of (conditional) models described by Altun and Smola (2006) and Dudík and Schapire (2006) at our disposition. They are characterized via

$$\underset{p}{\text{minimize}} -H(p) \text{ subject to } \|\mathbf{E}_{z\sim p}[\phi(z)] - \mu\| \leq \varepsilon.$$

Here p is a distribution, H is an entropy-like quantity defined on the space of distributions, and $\phi(z)$ is some evaluation map into a Banach space. This means that the optimization problem can be viewed as an approximate maximum entropy estimation problem, where we do not enforce exact moment matching of μ but rather allow ε slack. In both Altun and Smola (2006) and Dudík and Schapire (2006) the emphasis lay on *unconditional* density models: the dual of the above optimization problem. In particular, it follows that for H being the Shannon-Boltzmann entropy, the dual optimization problem is the maximum a posteriori estimation problem, which is what we are solving here.

In the conditional case, p denotes the collection of probabilities $p(y|x_i)$ and the operator $\mathbf{E}_{z\sim p}[\phi(z)] = \frac{1}{m}\sum_{i=1}^{m} \mathbf{E}_{y|p(y|x_i)}[\phi(x_i,y)]$ is the conditional expectation operator on the set of observations. Finally, $\mu = \frac{1}{m}\sum_{i=1}^{m}\phi(x_i,y_i)$, that is, it describes the empirical observations. We have two design parameters:

6.1.1 FUNCTION SPACE

Depending on which Banach Space norm we may choose to measure the deviation between μ and its expectation with respect to p in terms of e.g. the ℓ_2 norm, the ℓ_1 norm or the ℓ_∞ norm. The latter leads to sparse coding and convex combinations. This means that instead of solving an optimization problem of the form of (2) we would minimize expression of the form

$$\sum_{i=1}^{m} g(\theta|x_i) - m\langle \mu_{XY}, \theta\rangle + \lambda \|\theta\|^p_{\mathcal{B}^*},$$

where $p \geq 1$ and $\mathcal{B}^*$ is the Banach space of the natural parameter θ which is dual to the space $\mathcal{B}$ associated with the evaluation functionals $\phi(x,y)$. The most popular choice for $\mathcal{B}^*$ is ℓ_1 which leads to sparse coding (Candes and Tao, 2005; Chen et al., 1995).

6.1.2 ENTROPY AND REGULARITY

Depending on the choice of entropy and divergence functionals we obtain a range of diverse estimators. For instance, if we were to choose the *unnormalized* entropy instead of the entropy, we would obtain algorithms more akin to boosting. We may also use Csiszar and Bregmann divergences. The key point is that our reasoning of estimating μ_{XY} based on an aggregate of samples with unknown labels but known label proportions is still applicable.

6.2 Unknown Test Label Proportions

In many practical applications we may not actually know the label proportions on the test set. For instance, when deploying the algorithm to assess the spam in a user's mailbox we will *not* know

what the fraction would be. Nor is it likely that the user would be willing or able or trustworthy enough to provide a reliable estimate. This means that we need to estimate those proportions in addition to the class means μ_x^{class}.

We may use a fairly straightforward simplification of the covariate shift correction procedure of Huang et al. (2007) in this context. The basic idea is to exploit the fact that there the map $p(x) \to \mu[p(x)] = \mathbf{E}_x[\psi(x)]$ is injective for characteristic kernels (Sriperumbudur et al., 2008). Examples of such a characteristic kernel is Gaussian RBF, Laplacian, and B_{2n+1}-splines. This means that as long as the conditional distributions $p(x|y)$ are different for different choices of y we will be able to recover the test label proportions by the simple procedure of minimizing the distance between $\mu[p]$ and $\sum_y \alpha_y \mu[p(x|y)]$. While we may not have access to the true expectations we are still able to estimate $\mu_x^{\text{class}}[y]$ for all $y \in \mathcal{Y}$. This leads to the optimization problem

$$\begin{aligned} &\underset{\alpha}{\text{minimize}} \quad \left\| \frac{1}{m} \sum_{i=1}^{m} \psi(x_i) - \sum_{y \in \mathcal{Y}} \alpha_y \mu_X^{\text{class}}[y] \right\|^2 \\ &\text{subject to } \alpha_y \geq 0 \text{ and } \sum_{y \in \mathcal{Y}} \alpha_y = 1. \end{aligned} \tag{15}$$

Here the sum is taken over the elements of the test set, that is $x_j \in X$. Very similar bounds to those by Huang et al. (2007) can be obtained and they are omitted for the sake of brevity as the reasoning is essentially identical.

Note that obviously (15) may be used *separately* from the previous discussion, that is, when the training proportions are known but the test proportions are not. However, we believe that the most significant benefit is obtained in using both methods in conjunction since many practical situations exhibit both problems simultaneously.

7. Related Work and Alternatives

While being highly relevant in practice, the problem has not seen as much attention by researchers as one would expect. Some of the few works which cover a related subject are those by Chen et al. (2006) and Musicant et al. (2007), and by Kück and de Freitas (2005). We hope that our work will stimulate research in this area as relevant problems are fairly widespread.

7.1 Transduction

In transduction one attempts to solve a related problem: the patterns x_i on the test set are known, usually also some label proportions on the test set are known but obviously the actual labels on the test set are *not* known. One way of tackling this problem is to perform transduction by enforcing a proportionality constraint on the unlabeled data, e.g. via a Gaussian Process model (Gärtner et al., 2006; Mann and McCallum, 2007).

At first glance these methods might seem applicable for our problem but they do require that we have at least some labeled instances of *all classes* at our disposition which need to be drawn in an unbiased fashion. This is clearly not the case in our setting. That said, it is well possible to use our setting in the context of transduction, that is, to replace the unknown mean μ_{XY}^{test} on the test set by the empirical estimate on the training set. Such strategies lead to satisfactory performance on par with (albeit not exceeding) existing transduction approaches.

7.2 Self Consistent Proportions

Kück and de Freitas (2005) introduced a more informative variant of the binary multiple-instance learning, in which groups of instances are given along with estimates of the fraction of positively-labeled instances per group. The authors build a fully generative model of the process which determines the assignment of observations to individual bags. Such a procedure is likely to perform well when a large number of bags is present.

In order to deal with the estimation of the missing variables a MCMC sampling procedure is used. While Kück and de Freitas (2005) describe the approach only for a binary problem, it could be extended easily to multiclass settings.

In a similar vein, Chen et al. (2006) and Musicant et al. (2007) also use a self-consistent approach where the conditional class estimates need to match the observed ones. Consequently it shares the same similar drawbacks, since we typically only have as many sets as classes.

7.3 Conditional Probabilities

A seemingly valid alternative approach is to try building a classifier for $p(i|x)$ and subsequently recalibrating the probabilities to obtain $p(y|x)$, e.g. via $p(y|i)$. At first sight this may appear promising since this method is easily implemented by most discriminative methods. The idea would be to reconstruct $p(y|x)$ by

$$p(y|x) = \sum_i \pi_{iy} p(i|x).$$

However, this is not a useful estimator in our setting for a simple reason: it assumes the conditional independence $y \perp\!\!\!\perp x \mid i$, which obviously does not hold. Instead, we have the property that $i \perp\!\!\!\perp x| \, y$, that is, the distribution over x for a given class label does not depend on the bag. This mismatch in the probabilistic model can lead to disastrous estimates as the following simple example illustrates:

Example 1 *Assume that $\mathcal{X}, \mathcal{Y} = \{1,2\}$ and that $p(y=1|x=1) = p(y=2|x=2) = 1$. In other words, the estimation problem is solvable since the classes are well separated. Moreover, assume that π is given by*

$$\pi = \begin{bmatrix} 0.5-\varepsilon & 0.5+\varepsilon \\ 0.5 & 0.5 \end{bmatrix} \text{ for } 0 < \varepsilon \ll 1.$$

Here, $p(i|x)$ is useless for estimating $p(y|x)$, since we will only exceed random guessing by at most ε. On the other hand, it is easily possible to obtain a good estimate for μ_{XY} by our proposed procedure.

The reason for this failure can be found in the following expansion

$$p(y|x) = \sum_i p(y|x,i)p(i|x) \neq \sum_i p(y|i)p(i|x) \text{ since } p(y|x,i) \neq p(y|i). \tag{16}$$

The problem with (16) is that the estimator does not really attempt to compute the probability $p(y|x)$, which we are interested in but instead, it attempts to discern which mixture distribution p_i the observation x most likely originated from. For this to work we would need good probability estimates as the *basis* of reweighting. Our approach tackles the problem at the source by recalibrating the sufficient statistics directly.

7.4 Reduction to Binary

For binary classification and real-valued classification scores we may resort to a rather straightforward heuristic: build a classifier which is able to distinguish between the sets X_1 and X_2 and subsequently threshold labels such that the appropriate fraction of observations in X_1 and X_2 matches the proper labels. The intuition is that since the bags X_1 and X_2 do contain some information about how the two classes differ, we should be able to use this information to distinguish between different class labels.

It is likely that one might be able to obtain a proper reduction bound in this context. However, extensions to multi-class are highly nontrivial. It also turns out that even in the binary case this method, while overall fairly competitive, is inferior to our approach.

7.5 Density Estimation

One way of obtaining $p(x|i)$ is to carry out density estimation. While, in principle, this approach is flawed because of the incorrect conditional independence assumptions, it can still lead to acceptable results whenever each of the bags contains one majority class. This allows us to obtain

$$p(x|y) = \sum_i \left[\pi^{-1}\right]_{yi} p(x|i).$$

To re-calibrate the probability estimates Bayes' theorem is invoked to compute posterior probabilities. Since this approach involves density estimation it tends to fail fairly catastrophically for high-dimensional data due to the curse of dimensionality. These problems are also manifest in the experiments.

8. Experiments

Data Sets: We use binary and three-class classification data sets from the UCI repository[1] and the LibSVM site.[2] If separate training and test sets are available, we merge them before performing nested 10-fold cross-validation. Since we need to generate as many splits as classes, we limit ourselves to three classes.

For the binary data sets we use half of the data for X_1 and the rest for X_2. We also remove all instances of class 2 from X_1. That is, the conditional class probabilities in X_2 match those from the repository, whereas in X_1 their counterparts are deleted.

For three-class data sets we investigate two different partitions. In scenario A we use class 1 exclusively in X_1, class 2 exclusively in X_2, and a mix of all three classes weighted by $(0.5 \cdot p(1), 0.6 \cdot p(2), 0.7 \cdot p(3))$ to generate X_3. In scenario B we use the following splits

$$\begin{bmatrix} c_1 \cdot 0.4 \cdot p(1) & c_1 \cdot 0.2 \cdot p(2) & c_1 \cdot 0.2 \cdot p(3) \\ c_2 \cdot 0.1 \cdot p(1) & c_2 \cdot 0.2 \cdot p(2) & c_2 \cdot 0.1 \cdot p(3) \\ c_3 \cdot 0.5 \cdot p(1) & c_3 \cdot 0.6 \cdot p(2) & c_3 \cdot 0.7 \cdot p(3) \end{bmatrix}.$$

Here the constants c_1, c_2 and c_3 are chosen such that the probabilities are properly normalized. As before, X_3 contains half of the data.

1. UCI can be found at `http://archive.ics.uci.edu/ml/`.
2. LibSVM can be found at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/`.

Model Selection: As stated, we carry out a *nested* 10-fold cross-validation procedure: 10-fold cross-validation to assess the performance of the estimators; within each fold, 10-fold cross-validation is performed to find a suitable value for the parameters.

For supervised classification, i.e. discriminative sorting, such a procedure is quite straightforward because we can directly optimize for classification error. For kernel density estimation (KDE), we use the log-likelihood as our criterion.

Due to the high number of hyper-parameters (at least 8) in MCMC, it is difficult to perform *nested* 10-fold cross-validation. Instead, we choose the *best* parameters from a simple 10-fold crossvalidation run. In other words, we are giving the MCMC method an unfair advantage over our approach by reporting the best performance during the model selection procedure.

Finally, for the re-calibrated sufficient statistics $\hat{\mu}_{XY}$ we use the estimate of the log-likelihood on the validation set as the criterion for cross-validation, since no other quantity, such as classification errors is readily available for estimation.

Algorithms: For discriminative sorting we use an SVM with a Gaussian RBF kernel whose width is set to the median distance between observations (Schölkopf, 1997); the regularization parameter is chosen by cross-validation. The same strategy applies for our algorithm. For KDE, we use Gaussian kernels. Cross-validation is performed over the kernel width. For MCMC, 10 000 samples are generated after a burn-in period of 10 000 steps (Kück and de Freitas, 2005).

Optimization: Bundle methods (Smola et al., 2007; Teo et al., 2007) are used to solve the optimization problem in Algorithm 1. For our regularized log-likelihood, the solver converges to ε precision in $O(\log(1/\varepsilon))$ steps.

Results: The experimental results are summarized in Table 3. Our method outperforms KDE and discriminative sorting. In terms of computation, our approach is somewhat more efficient, since it only needs to deal with a smaller sample size (only X rather than the union of all X_i). The training time for our method is less than 2 minutes for all cases, whereas MCMC on average takes 15 minutes and maybe even much longer when the number of active kernels and/or observations are high. Note that KDE fails on two data sets due to numerical problems (high dimensional data).

Our method also performs well on multiclass data sets. As described in Section 5.2, the quality of our minimizer of the negative log-posterior depends on the mixing matrix and this is noticeable in the reduction of performance for the dense mixing matrix (scenario B) in comparison to the better conditioned sparse mixing matrix (scenario A). In other words, for ill conditioned π even our method has its limits, simply due to numerical considerations of effective sample size.

Unknown test label proportions: In this experiment, we use binary and three-class classification data sets with the same split procedure as in the previous experiment but we select testing examples by a biased procedure to introduce unknown test label proportions. To describe our biased procedure, consider a random variable ξ_i for each point in the pool of possible testing samples where $\xi_i = 1$ means the i-th sample is being included and $\xi_i = 0$ means the sample is discarded. In our case, the biased procedure only depends on the label y, i.e. $P(\xi = 1|y = 1) = 0.5$ and $P(\xi = 1|y = -1) = 1.0$ for binary problems and $P(\xi = 1|y = 1) = 0.6$, $P(\xi = 1|y = 2) = 0.3$, and $P(\xi = 1|y = 3) = 0.1$ for three-class problems. We then estimate the test proportion by solving the quadratic program in (15) with interior point methods (or any other successive optimization procedure). Since we are interested particularly to assess the effectiveness of our test proportion estimation method, in solving (15) we assume that we can compute $\mu_X^{\text{class}}[y]$ directly, i.e. the instances are labeled. The mean square error rates of test proportions for several binary and three-class data

Data	MM	KDE	DS	MCMC	BA
ionosphere	18.4±3.2	**17.5±3.2**	**12.2±2.6**	18.0±2.1	35.8
iris	**10.0±3.6**	**16.8±3.4**	**15.4±1.1**	**21.1±3.6**	29.9
optdigits	1.8±0.5	**0.7±0.4**	9.8±1.2	2.0±0.4	49.1
pageblock	**3.8±2.3**	7.1±2.8	18.5±5.6	**5.4±2.8**	43.9
pima	**27.5±3.0**	34.8±0.6	34.4±1.7	**23.8±1.8**	34.8
tic	31.0±1.5	34.6±0.5	**26.1±1.5**	31.3±2.5	34.6
yeast	**9.3±1.5**	**6.5±1.3**	25.6±3.6	10.4±1.9	39.9
wine	**7.4±3.0**	**12.1±4.4**	18.8±6.4	**8.7±2.9**	40.3
wdbc	**7.8±1.3**	**5.9±1.2**	10.1±2.1	15.5±1.3	37.2
sonar	**24.2±3.5**	35.2±3.5	31.4±4.0	39.8±2.8	44.5
heart	**30.0±4.0**	38.1±3.8	**28.4±2.8**	**33.7±4.7**	44.9
breastcancer	**5.3±0.8**	14.2±1.6	**3.5±1.3**	**4.8±2.0**	34.5
australian	**17.0±1.7**	33.8±2.5	**15.8±2.9**	30.8±1.8	44.4
svmguide3	**20.4±0.9**	27.2±1.3	25.5±1.5	24.2±0.8	23.7
adult	**18.9±1.2**	24.5±1.3	22.1±1.4	**18.7±1.2**	24.6
cleveland	**19.1±3.6**	35.9±4.5	**23.4±2.9**	**24.3±3.1**	22.7
derm	**4.9±1.4**	27.4±2.6	**4.7±1.9**	14.2±2.8	30.5
musk	**25.1±2.3**	28.7±2.6	**22.2±1.8**	**19.6±2.8**	43.5
german	**32.4±1.8**	41.6±2.9	37.6±1.9	**32.0±0.6**	32.0
covertype	37.1±2.5	41.9±1.7	**32.4±1.8**	41.1±2.2	45.9
splice	**25.2±2.0**	35.5±1.5	**26.6±1.7**	28.8±1.6	48.4
gisette	**10.3±0.9**	†	**12.2±0.8**	50.0±0.0	50.0
madelon	**44.1±1.5**	†	**46.0±2.0**	49.6±0.2	50.0
cmc	**37.5±1.4**	43.8±0.7	45.1±2.3	46.9±2.6	49.9
bupa	**48.5±2.9**	50.8±5.1	**40.3±4.9**	50.4±0.8	49.7
protein A	**43.3±0.4**	48.9±0.9	N/A	65.5±1.7	60.6
protein B	**46.9±0.3**	55.2±1.5	N/A	66.1±2.1	60.6
dna A	**14.8±1.2**	28.1±0.6	N/A	39.8±2.6	41.6
dna B	**31.3±1.3**	**30.4±0.7**	N/A	41.5±0.1	41.6
senseit A	**19.8±0.1**	44.2±0.0	N/A	‡	44.2
senseit B	**21.1±0.1**	44.2±0.0	N/A	‡	44.2

Table 3: Classification error on UCI/LibSVM data sets. Errors are reported in mean ± standard error. The best result and those not significantly worse than it, are highlighted in boldface. We use a one-sided paired t-test with 95% confidence. `MM`: Mean Map (our method); `KDE`: Kernel Density Estimation; `DS`: Discriminative Sorting (only applicable for binary classification); `MCMC`: the sampling method; `BA`: Baseline, obtained by predicting the major class. †: Program fails (too high dimensional data - only `KDE`). ‡: Program fails (large data sets - only `MCMC`).

sets are presented in Table 4. The results show that our proportion estimation method works reasonably well.

Overdetermined systems: Here we are interested to assess the performance of our estimator with optimized weights when we have more data sets n than class labels $|\mathcal{Y}|$ with varying number of observations m_i per data set. We simulate the problem in binary settings with the following split ($n = 8$)

$$\begin{bmatrix} c_1 \cdot 0.25 \cdot p(1) & c_1 \cdot 0.10 \cdot p(2) \\ c_2 \cdot 0.15 \cdot p(1) & c_2 \cdot 0.10 \cdot p(2) \\ c_3 \cdot 0.05 \cdot p(1) & c_3 \cdot 0.20 \cdot p(2) \\ c_4 \cdot 0.05 \cdot p(1) & c_4 \cdot 0.10 \cdot p(2) \\ c_5 \cdot 0.05 \cdot p(1) & c_5 \cdot 0.00 \cdot p(2) \\ c_6 \cdot 0.05 \cdot p(1) & c_6 \cdot 0.05 \cdot p(2) \\ c_7 \cdot 0.05 \cdot p(1) & c_7 \cdot 0.15 \cdot p(2) \\ c_8 \cdot 0.35 \cdot p(1) & c_8 \cdot 0.30 \cdot p(2) \end{bmatrix}$$

and the split ($n = 6$) in three-class settings is as follows

$$\begin{bmatrix} c_1 \cdot 0.30 \cdot p(1) & c_1 \cdot 0.10 \cdot p(2) & c_1 \cdot 0.00 \cdot p(3) \\ c_2 \cdot 0.10 \cdot p(1) & c_2 \cdot 0.10 \cdot p(2) & c_2 \cdot 0.20 \cdot p(3) \\ c_3 \cdot 0.05 \cdot p(1) & c_3 \cdot 0.00 \cdot p(2) & c_3 \cdot 0.05 \cdot p(3) \\ c_4 \cdot 0.05 \cdot p(1) & c_4 \cdot 0.20 \cdot p(2) & c_4 \cdot 0.05 \cdot p(3) \\ c_5 \cdot 0.00 \cdot p(1) & c_5 \cdot 0.05 \cdot p(2) & c_5 \cdot 0.10 \cdot p(3) \\ c_6 \cdot 0.50 \cdot p(1) & c_6 \cdot 0.55 \cdot p(2) & c_6 \cdot 0.60 \cdot p(3) \end{bmatrix}.$$

We use BFGS to obtain the optimal weights of the minimization problem in (11). We perform 10-fold cross validation with respect to the log-likelihood. The error rates are presented in Table 5. For all cases except one, the estimator with optimized weights improves error rates compared with the unweighted one.

Binary data sets

Data	**MSE**
australian	0.00804±0.00275
breastcancer	0.00137±0.00063
adult	0.00610±0.00267
derm	0.00398±0.00175
gisette	0.00331±0.00108
wdbc	0.00319±0.00103

Three-class data sets

Data	**MSE**
protein	0.00290±0.00066
dna	0.00339±0.00075
senseit	0.00072±0.00031

Table 4: Unknown test label proportion case. Square errors of estimating the test proportions on UCI/LibSVM data sets. The 10-run errors are reported in mean ± standard error.

Stability of Mixing Matrices: Lastly, we are interested to assess the performance of our proposed method when the given mixing matrix π are perturbed so that they do not exactly match how the data is generated. We used binary classification data sets and defined the perturbed mixing

Binary data sets

Data	unweighted	weighted
wdbc	23.29±2.68	**14.22±1.79**
australian	34.44±4.03	29.58±3.71
svmguide3	24.28±2.20	**18.50±1.73**
gisette	8.77±1.05	7.69±0.51
splice	33.43±1.65	**21.12±2.59**

Three-class data sets

Data	unweighted	weighted
protein	57.46±0.02	57.46±0.02
senseit	28.25±2.60	23.51±0.78
dna	20.01±1.26	**16.80±1.19**

Table 5: Overdetermined systems. Errors of weighted/unweighted estimators for overdetermined systems on UCI/LibSVM data sets. The 10-fold cross validation errors are reported in mean ± standard error. The numbers in boldface are significant with 95% confidence (one-sided paired t-test).

matrix as

$$\tilde{\pi} = \pi + \Delta = \begin{bmatrix} 1 & 0 \\ \rho & 1-\rho \end{bmatrix} + \begin{bmatrix} -\varepsilon_1 & \varepsilon_1 \\ \varepsilon_2 & -\varepsilon_2 \end{bmatrix}.$$

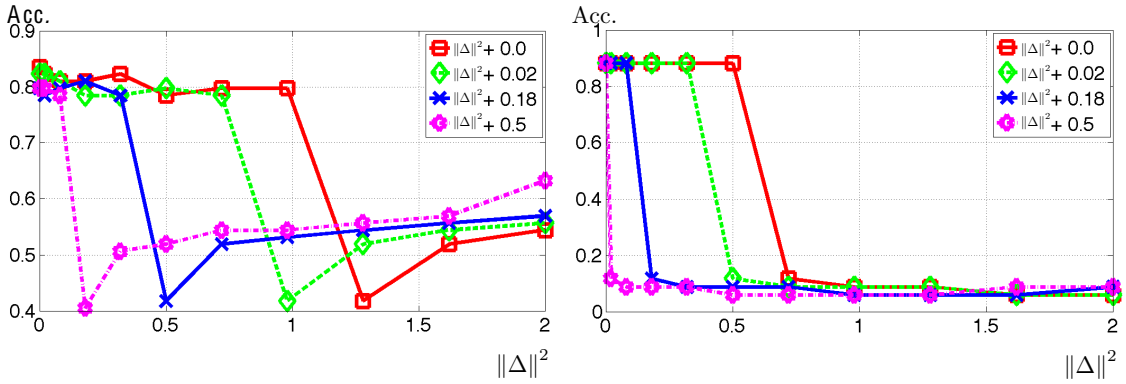
We varied $\varepsilon_1 \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ and $\varepsilon_2 \in \{0.0, 0.1, 0.3, 0.5\}$ and measured the performance as a function of the size of the perturbation, $\eta = \|\Delta\|^2 = tr(\Delta^\top \Delta)$. Note that unperturbed mixing matrix refer to the case of $\{\varepsilon_1, \varepsilon_2\} = \{0, 0\}$. The experiments are summarized in Figure 2. The results suggest that for a reasonable size of perturbations, our method is stable.

9. Conclusion

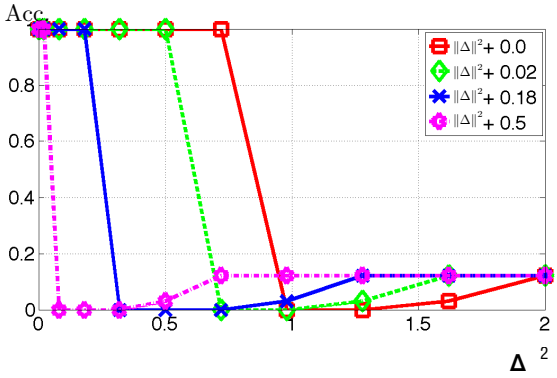
In this paper we obtained a rather surprising result, namely that it is possible to consistently reconstruct the labels of a data set if we can only obtain information about the proportions of occurrence of each class (in at least as many data aggregates as there are classes). In particular, we proved that up to constants, our algorithm enjoys the same rates of convergence afforded to methods which have full access to all label information.

This finding has significant implications with regard to the amount of privacy afforded by summary statistics. In particular, it implies that whenever accurate summary statistics exist and whenever the available individual statistics are highly dependent on the summarized random variable we will be able to perform inference on the summarized variable with a high degree of confidence. In other words, some techniques used to anonymize observations, e.g. demographic data, may not be really safe (at least when it is possible to estimate the missing information, provided enough data).

Recently Chiaia et al. (2007) applied a summarization technique to infer drug use based on the concentration of metabolites in the sewage of cities, suburbs or at an even more finely grained resolution. While this only provides aggregate information about the proportions of drug users, such data, in combination with detailed demographic information might be used to perform more detailed inference with regard to the propensity of individuals to use controlled substances. It is in these types of problem where our method could be applied straightforwardly.



(a) Adult (b) Australian



(c) Breastcancer

Figure 2: Performance accuracy of binary classification data sets ($n = |\mathcal{Y}| = 2$) as a function of the amount of perturbation applied to the mixing matrix, $\|\Delta\|^2 = tr(\Delta^\top \Delta)$ with $\Delta = \tilde{\pi} - \pi$. 2(a): Adult, 2(b): Australian and 2(c): Breastcancer data sets. x-axis denotes $\|\Delta\|^2$ as a function of $\varepsilon_1 \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Color coded plots denote $\|\Delta\|^2$ as a function of $\varepsilon_2 \in \{0.0, 0.1, 0.3, 0.5\}$, for example red colored plot refers to performance when only label proportions of the first set are perturbed.

Acknowledgments

We thank Hendrik Kück and Choon Hui Teo for providing us with optimization code. NICTA is funded by the Australian Government's Backing Australia's Ability and the Centre of Excellence programs. This work was supported by the PASCAL network of excellence of the European Union.

References

Y. Altun and A.J. Smola. Unifying divergence minimization and statistical inference via convex duality. In H.U. Simon and G. Lugosi, editors, *Proc. Annual Conf. Computational Learning Theory*, LNCS, pages 139–153. Springer, 2006.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482, 2002.

E. Candes and T. Tao. Decoding by linear programming. *IEEE Trans. Info Theory*, 51(12):4203–4215, 2005.

B.C. Chen, L. Chen, R. Ramakrishnan, and D.R. Musicant. Learning from aggregate views. In L. Liu, A. Reuter, K.Y. Whang, and J. Zhang, editors, *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, pages 3–12, Atlanta, GA, 2006.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. Technical Report 479, Department of Statistics, Stanford University, May 1995.

A. C. Chiaia, C. Banta-Green, L. Power, D. L. Sudakin, and J. A. Field. Community burdens of methamphetamine and other illicit drugs. In *6th International Conference on Pharmaceuticals and Enocrine Disrupting Chemicals in Water*, 2007.

M. Dudík and R. E. Schapire. Maximum entropy distribution estimation with generalized regularization. In Gábor Lugosi and Hans U. Simon, editors, *Proc. Annual Conf. Computational Learning Theory*. Springer Verlag, June 2006.

T. Gärtner, Q.V. Le, S. Burton, A.J. Smola, and S.V.N. Vishwanathan. Large-scale multiclass transduction. In *Neural Information Processing Systems*, pages 411–418. MIT Press, 2006.

T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. Technical Report 156, Max-Planck-Institut für biologische Kybernetik, 2006. To appear in the Annals of Statistics.

J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems 19*, Cambridge, MA, 2007. MIT Press.

H. Kück and N. de Freitas. Learning about individuals from group statistics. In *Uncertainty in Artificial Intelligence (UAI)*, pages 332–339, Arlington, Virginia, 2005. AUAI Press.

M. Ledoux and M. Talagrand. *Probability in Banach Spaces*. Springer, 1991.

G. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In Zoubin Ghahramani, editor, *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007),Corvallis, OR*, pages 593–600. Omnipress, 2007.

S. Mendelson. Rademacher averages and phase transitions in glivenko-cantelli classes. *IEEE Trans. Inform. Theory*, 48(1):251–263, 2002.

D.R. Musicant, J. Christensen, and J.F. Olson. Supervised learning by training on aggregate outputs. In *IEEE International Conference on Data Mining*, 2007.

N. Quadrianto, A. Smola, T. Caetano, and Q. Le. Estimating labels from label proportions. In W. Cohen, A. McCallum, and S. Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 776–783. Omnipress, 2008.

B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, Munich, 1997. Download: http://www.kernel-machines.org.

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

A.J. Smola, S. V. N. Vishwanathan, and Q.V. Le. Bundle methods for machine learning. In Daphne Koller and Yoram Singer, editors, *Advances in Neural Information Processing Systems 20*, Cambridge MA, 2007. MIT Press.

B. Sriperumbudur, A. Gretton, K. Fukumizu, G. Lanckriet, and B. Schölkopf. Injective hilbert space embeddings of probability measures. In *Proceedings of the 21st Annual Conference on Learning Theory*, pages 111–122, 2008.

C.H. Teo, Q. Le, A.J. Smola, and S.V.N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD)*. ACM, 2007.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005.

P.Å. Wedin. Perturbation theory for pseudo-inverses. *BIT Numerical Mathematics*, 13(2), 1973.

Journal of Machine Learning Research 10 (2009) 2375-2411 Submitted 2/08; Revised 4/09; Published 10/09

Exploiting Product Distributions to Identify Relevant Variables of Correlation Immune Functions

Lisa Hellerstein HSTEIN@CIS.POLY.EDU
Department of Computer Science and Engineering
Polytechnic Institute of NYU
Brooklyn, NY 11201, USA

Bernard Rosell BROSELL@ATT.COM
AT&T Laboratories
200 South Laurel Ave.
Middletown, NJ 07748, USA

Eric Bach BACH@CS.WISC.EDU
Department of Computer Sciences
University of Wisconsin, Madison
Madison, WI 53706, USA

Soumya Ray SRAY@EECS.CASE.EDU
Department of Electrical Engineering and Computer Science
Case Western Reserve University
Cleveland, OH 44106, USA

David Page PAGE@BIOSTAT.WISC.EDU
Department of Biostatistics and Medical Informatics
University of Wisconsin, Madison
Madison, WI 53706, USA

Editor: Peter Bartlett

Abstract

A Boolean function f is *correlation immune* if each input variable is independent of the output, under the uniform distribution on inputs. For example, the parity function is correlation immune. We consider the problem of identifying relevant variables of a correlation immune function, in the presence of irrelevant variables. We address this problem in two different contexts. First, we analyze *Skewing*, a heuristic method that was developed to improve the ability of greedy decision tree algorithms to identify relevant variables of correlation immune Boolean functions, given examples drawn from the uniform distribution (Page and Ray, 2003). We present theoretical results revealing both the capabilities and limitations of skewing. Second, we explore the problem of identifying relevant variables in the *Product Distribution Choice* (PDC) learning model, a model in which the learner can choose product distributions and obtain examples from them. We prove a lemma establishing a property of Boolean functions that may be of independent interest. Using this lemma, we give two new algorithms for finding relevant variables of correlation immune functions in the PDC model.

Keywords: correlation immune functions, skewing, relevant variables, Boolean functions, product distributions

1. Introduction

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *correlation immune* if for every input variable x_i, the values of x_i and $f(x_1,\ldots,x_n)$ are independent, with respect to the uniform distribution on $\{0,1\}^n$ (cf. Roy, 2002). Examples of correlation immune functions include parity of $k \geq 2$ variables, the constant functions $f \equiv 1$ and $f \equiv 0$, and the function $f(x) = 1$ iff all bits of x are equal.

If a function f is not correlation immune, then given access to examples of f drawn from the uniform distribution, one can easily identify (at least one) relevant variable of f by finding an input variable that is correlated with the output of f. This approach clearly fails if f is correlation immune.

We consider the problem of identifying relevant variables of a correlation immune function, in the presence of irrelevant variables. This problem has been addressed by machine learning practitioners through the development of heuristics, and by computational learning theorists, who have analyzed the problem in standard learning models. We were motivated by work from both communities, and present results related to both types of work. First, we present a theoretical analysis of *skewing*, a heuristic method that was developed to improve the ability of greedy decision tree learning algorithms to identify relevant variables of correlation immune functions, given examples drawn from the uniform distribution (Page and Ray, 2003; Ray and Page, 2004). Second, we present algorithms for identifying relevant variables in the *Product Distribution Choice* (PDC) model of learning. The PDC model, which we introduce below, is a variant of the standard PAC learning model (Valiant, 1984) in which the learner can specify product distributions and sample from them.[1]

Greedy decision tree learning algorithms perform poorly on correlation immune functions because they rely on measures such as Information Gain (Quinlan, 1997) and Gini gain (Breiman et al., 1984) to choose which variables to place in the nodes of the decision tree. The correlation immune functions are precisely those in which every attribute has zero gain under all standard gain measures, when the gain is computed on the complete data set (i.e., the truth table) for the function. Thus when examples of a correlation immune function are drawn uniformly at random from the complete data set, the learning algorithms have no basis for distinguishing between relevant and irrelevant attributes.

Experiments have shown skewing to be successful in learning many correlation immune functions (Page and Ray, 2003). One of the original motivations behind skewing was the observation that obtaining examples from non-uniform product distributions can be helpful in learning particular correlation immune functions such as parity. Skewing works by reweighting the given training set to simulate receiving examples from a subclass of product distributions called *skewed* distributions. In a skewed distribution, each input variable x_i is independently set to 1 with probability p_i; further, there is a fixed probability p, such that each p_i is either equal to p or to $1-p$.

However, simulating alternative distributions is not the same as sampling directly from them. The *Product Distribution Choice* (PDC) model allows such direct sampling. This model can be seen as a variant of the PAC model, and has similarities with other learning models studied previously (see Section 5). In the PDC model, the learner has access to an oracle from which it can request examples. Before requesting an example, the learner specifies a product distribution. The oracle then supplies an example drawn from that distribution. In our study of the PDC model, we focus

1. Our PDC model algorithms could be presented independently of any discussion of the skewing heuristic. However, the algorithms rely on technical results that we originally proved to analyze skewing, and we present those technical results as part of our discussion of skewing. Readers who are only interested in understanding the PDC algorithms will need to read some of the material on skewing, but can skip Sections 9.3 and 11.

on a fundamental learning task: the problem of identifying relevant variables in the presence of irrelevant ones.

Note that by setting the parameters of the product distribution to be equal to 0 and 1, one can simulate membership queries in the PDC model. However, we are particularly interested in exploring learning in the PDC model when the parameters of the chosen product distributions are bounded away from 0 and 1.

Our interest in the PDC model is motivated not just by our study of skewing, but by a more general question: In learning, how much does it help to have access to data from different distributions? In practice, it may be possible to obtain data from different distributions by collecting it from different sources or populations. Alternatively, one may be able to alter environmental conditions to change the distribution from which data is obtained. In such settings, it can be expensive to sample from too many distributions, and it may be difficult or impossible to sample from "extreme" distributions. Thus in the PDC model, we are concerned not just with time and sample complexity, but also in the number and type of product distributions specified.

2. Summary of Results

We begin by showing that, given a complete data set, skewing will succeed. That is, given the complete truth table of a target Boolean function as the training set, skewing will find a relevant variable of that function. (More particularly, under any random choice of skewing parameters, a single round of the skewing procedure will find a relevant variable with probability 1.) This result establishes that the approach taken by skewing is fundamentally sound. However, it says nothing about the effectiveness of skewing when, as is typically the case, the training set contains only a small fraction of the examples in the truth table. In particular, this result does not address the question of whether skewing would be effective given only a polynomial-size sample and polynomial time.

We also analyze a variant of skewing called *sequential skewing* (Ray and Page, 2004), in the case that the full truth table is given as input. Experiments indicate that sequential skewing scales better to higher dimensional problems than standard skewing. We show here, however, that even when the entire truth table is available as the training set, sequential skewing is ineffective for a subset of the correlation immune functions known as the *2-correlation immune* functions. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is 2-correlation immune if, for every pair of distinct input variables x_i and x_j, the variables x_i, x_j, and $f(x_1,\ldots,x_n)$ are mutually independent. Thus, any practical advantage sequential skewing has over standard skewing comes at the cost of not working on this subset of functions.

We present two new algorithms in the PDC model for identifying a relevant variable of an n-variable Boolean function with r relevant variables. The first algorithm uses only r distinct p-biased distributions (i.e., distributions in which each input variable is independently set to 1 with some fixed probability p). It runs in time polynomial in n and its sample size, which is $O((r+1)^{2r} \ln \frac{2nr}{\delta})$. (The algorithm is randomized, but we also give a deterministic version achieving slightly different bounds.) The second algorithm uses $O(e^{3r} \ln \frac{1}{\delta})$ p-biased distributions, and runs in time polynomial in n and the sample size, $O(e^{9r}(r + \ln \frac{n}{\delta}) \ln(\frac{1}{\delta}))$. Both algorithms choose the distributions they use non-adaptively. For $r = O(\log n)$, only the second algorithm runs in time polynomial in n, but the first uses $O(\log n)$ distributions, whereas the second uses a number of distributions that depends polynomially on n.

The second of our two algorithms is based on a new sample complexity result that we prove using martingales.

Previous algorithms for identifying relevant variables in the PDC model, and achieving bounds similar to ours, use distributions that are not p-biased, and choose the distributions they use adaptively. Independently, Arpe and Mossel (to appear) have recently developed an algorithm that is similar to our first algorithm. We discuss these related algorithms further in Sections 5 and 10.

Since p-biased distributions are skewed distributions, our algorithms can be viewed as skewing algorithms for a setting in which it is possible to sample directly from skewed distributions, rather than to just simulate those distributions.

We also examine skewing in the context for which it was originally designed: learning from a random sample drawn from the uniform distribution. We prove a negative result in this context, a sample complexity lower bound for the problem of learning parity functions. Technically, we prove the bound for a variant of skewing, called skewing with independent samples, that is more amenable to analysis than standard skewing. For intuitive reasons, and based on experimental evidence, we think it likely that the bound also holds for standard skewing. The bound implies that skewing with independent samples requires a sample of size at least $n^{\Omega(\log n)}$ to find (with constant probability of failure) a relevant variable of an n-variable Boolean function computing the parity of $\log n$ of its variables.

Correlation immunity is defined in terms of the uniform distribution. We discuss a natural extension of correlation immunity to non-uniform product distributions. We give a simple example of a function that is correlation immune with respect to a non-uniform product distribution. Thus while functions like parity are difficult for greedy learners when examples come from the uniform distribution, other functions can be difficult when examples come from another product distribution.

Our analysis of skewing given a complete data set, and our two new algorithms in the PDC model, are both based on a lemma that we prove which shows that Boolean functions have a certain useful property. Specifically, we show that every non-constant Boolean function f on $\{0,1\}^n$ has a variable x_i such that induced functions $f_{x_i \leftarrow 0}$ and $f_{x_i \leftarrow 1}$ on $\{0,1\}^{n-1}$ (produced by hardwiring x_i to 0 and 1) do not have the same number of positive examples of Hamming weight k, for some k. This lemma may be of independent interest.

3. Organization of the Paper

We first give some background on skewing in Section 4. In Section 5, we discuss related work. Section 6 contains basic definitions and lemmas, including characterizations of correlation immune functions, and simple lemmas on quantities such as Gini gain and the magnitude of the first-order Fourier coefficients. It also contains a simple example of a function that is correlation immune with respect to a non-uniform product distribution. Section 7 discusses sample complexity bounds used later in the paper, and proves an upper bound on the estimation of Gini gain, based on martingales.

In Section 8, we prove the lemma showing the useful property of Boolean functions.

We begin our analysis of skewing in Section 9 with results for the setting in which the entire truth table is given as the training set.

Section 10 contains our two new algorithms for the PDC model. It also contains a discussion of two PDC algorithms that are implicit in the literature.

Finally, Section 11 contains our sample complexity lower bounds on learning parity functions using skewing with independent samples.

4. Background on Skewing

As a motivating example, suppose we have a Boolean function $f(x_1,\ldots,x_n)$ whose value is the parity of r of its variables. Function f is correlation immune. With respect to the uniform distribution on the domain of f, all n variables of f have zero gain. Equivalently, the first-order Fourier coefficients of f are all zero (cf. Section 6.3). But, with respect to other product distributions on the examples, the r relevant variables of f have non-zero gain, while the $n-r$ irrelevant variables still have zero gain (see Page and Ray, 2003; Arpe and Reischuk, 2007). This suggests that learning correlation immune functions might be easier if examples could be obtained from non-uniform product distributions.

In many machine learning applications, however, we have little or no control over the distribution from which we obtain training data. The approach taken by skewing is to reweight the training data, to simulate receiving examples from another distribution. More particularly, the skewing algorithm works by choosing a "preferred setting" (either 0 or 1) for every variable x_i in the examples, and a weighting factor p where $\frac{1}{2} < p < 1$. These choices define a product distribution over examples $x \in \{0,1\}^n$ in which each variable x_i has its preferred setting with probability p, and the negation of that setting with probability $1-p$.

To simulate receiving examples from this product distribution, the skewing algorithm begins by initializing the weight of every example in the training set to 1. Then, for each x_i, and each example, it multiplies the weight of the example by p if the value of x_i in the example matches its preferred setting, and by $1-p$ otherwise. This process is called "skewing" the distribution. The algorithm computes the gain of each variable with respect to the reweighting. The algorithm repeats this procedure a number of times, with different preferred settings chosen each time. Finally, it uses all the calculated gains to determine which variable to output. The exact method used varies in different skewing implementations. In the paper that introduced skewing, the variable chosen was the one whose calculated gains exceeded a certain threshold the maximum number of times (Page and Ray, 2003).

In the context of decision tree learning, skewing is applied at every node of the decision tree, in place of standard gain calculations. After running skewing on the training set at that node, the variable chosen by the skewing procedure is used as the split variable at that node.

In investigating skewing, we are particularly interested in cases in which the number of relevant variables is much less than the total number of variables. Optimally, we would like sample complexity and running time to depend polynomially on n and 2^r (and on $\log\frac{1}{\delta}$), so that we have a polynomial-time algorithm when $r = O(\log n)$.

5. Related Work

Throughout this paper, we focus on the problem of finding a relevant variable of a target Boolean function, given a labeled sample drawn from the uniform distribution. Given a procedure that finds a single relevant variable x_i of a Boolean function f (for any f with at most r relevant variables), it is usually easy to extend the procedure to find all relevant variables of the target by recursively applying it to the induced functions obtained by hardwiring x_i to 1 and 0 respectively.

It is a major open problem whether there is a polynomial-time algorithm for finding relevant variables of a Boolean function of $\log n$ relevant variables (out of n total variables) using examples from the uniform distribution (cf. Blum, 2003). Mossel et al. (2003) gave an algorithm for learning

arbitrary functions on r relevant variables, using examples drawn from the uniform distribution, in time polynomial in n^{cr} and $\ln(1/\delta)$, for some $c < 1$. This improves on the naïve algorithm which requires time polynomial in n^r for small r. The heart of the algorithm is a procedure to find a relevant variable. The algorithm of Mossel et al. uses both Gaussian elimination and estimates of Fourier coefficients, and is based on structural properties of Boolean functions.

Mossel et al. also briefly considered the question of finding a relevant variable, given examples drawn from a single product distribution $[p_1, \ldots, p_n]$.[2] They stated a result that is similar to our Theorem 9.1, namely that if a product distribution is chosen at random, then with probability 1, the Fourier coefficient (for that distribution) associated with any relevant variable will be non-zero. The important difference between that result and Theorem 9.1 is that our theorem applies not to all random product distributions, but just to random skewed distributions. Since skewed distributions have measure zero within the space of all product distributions, the result of Mossel et al. does not imply anything about skewed distributions.

In interesting recent work that was done independently of this paper, Arpe and Mossel (to appear) addressed the problem of finding relevant variables of a Boolean function, using examples from biased distributions. If an input to a Boolean function f is drawn from a p-biased distribution, the output of f on that input is a random variable. Arpe and Mossel observed that the expectation of this random variable is a polynomial in the bias, and expressed the Maclaurin series for this polynomial in terms of the Fourier coefficients of f. They used this expression to develop a family of algorithms for identifying relevant variables. For a function with r relevant variables, the s-th algorithm estimates Fourier coefficients of Hamming weight up to s, using about r/s distributions. They also extended their algorithms to allow estimation of biases by sampling, a problem we do not address here.

Applying the results of Arpe and Mossel for $s = 1$ to the case of uniformly spaced biases yields an algorithm that is almost the same as our first algorithm, with a very different correctness proof. Although Arpe and Mossel did not give the sample size of their algorithm explicitly, some computations show that it is larger than the sample size we give (in Theorem 10.1), by a factor roughly equal to 16^r. Like us, they used a large deviation bound to derive a sample size, but they did not estimate parameters for this bound in the best way known. If that is done, following the approach of Furst et al. (1991), the discrepancy vanishes.

The problem of learning parity functions has been extensively studied in various learning models. It is a well-known open question whether it is possible to PAC-learn parity functions in polynomial time, using examples drawn from the uniform distribution, in the presence of random classification noise. This problem is at least as difficult as other open problems in learning; in fact, a polynomial time algorithm for this problem would imply a polynomial-time algorithm for the problem mentioned above, learning functions of $\log n$ relevant variables using examples from the uniform distribution (Feldman et al., 2006).

Our lower bound result for parity in Section 11 relies on Fourier-based techniques previously used to prove lower bounds for learning parity in statistical query (SQ) learning learning models (Blum et al., 1994; Jackson, 2003). Roughly speaking, statistical query learning algorithms learn a target function by adaptively specifying predicates that are defined over labeled examples of the

2. They also claimed that this result implies an algorithm for learning functions with r relevant variables in time polynomial in 2^r, n, and $\ln(1/\delta)$, given examples drawn from almost any product distribution. However, the justification for their claim was faulty, since it does not take into account the magnitude of the non-zero Fourier coefficient.

function. For each such predicate, the algorithm obtains an estimate (within a certain tolerance) of the probability that a random labeled example of the function satisfies the given predicate.

Jackson (2003) proved that that any "SQ-based" algorithm for learning the class of all parity functions takes time $\Omega(2^{n/2})$. Jackson also showed that a more complex argument could be used to prove a stronger bound of $\Omega(2^n)$. For the problem of learning just the parity functions having r relevant variables, rather than all parity functions, these bounds become $\Omega(\binom{n}{r}^{1/2})$ and $\Omega(\binom{n}{r})$ respectively. Although skewing with independent samples is not an SQ-based algorithm, we prove a bound that is similar to the weaker of these two bounds, using a similar technique. (Our bound is for identifying a single relevant variable of the target parity function, rather than for learning the function.) The proof of Jackson's stronger bound relies on properties of SQ-based algorithms that are not shared by skewing with independent samples, and it is an open question whether a similar bound is achievable for skewing with independent samples.

Subsequent to Jackson's work, Yang gave lower bounds for learning parity using "honest" statistical queries (Yang, 2001, 2005). While the gain estimates performed in skewing seem to correspond to honest statistical queries, the correspondence is not direct. One cannot determine the gain of a variable with respect to a skewed distribution by using only a single honest statistical query. Because lower bounds in statistical query models rely on the fact that only limited information can be obtained from the examples in the sample used to answer a single query, lower bounds for learning with honest statistical queries do not directly imply lower bounds for skewing with independent samples. Further, we were unable to verify relevant lower bounds given by Yang.[3]

At the other extreme from correlation-immune functions are functions for which all first order Fourier coefficients are non-zero (i.e., all relevant variables have non-zero gain). This is true of monotone functions (see Mossel et al., 2003). Arpe and Reischuk, extending previous results, gave a Fourier-based characterization of the class of functions that can be learned using a standard greedy covering algorithm (Arpe and Reischuk, 2007; Akutsu et al., 2003; Fukagawa and Akutsu, 2005). This class is a superset of the set of functions for which all relevant variables have non-zero degree-1 Fourier coefficients.

The PDC model investigated in this paper has some similarity to the extended statistical query model of Bshouty and Feldman (2002). In that model, the learner can specify a product distribution in which each variable is set to 1 with probability $\rho, 1/2$ or $1-\rho$, for some constant $1/2 > \rho > 0$. The learner can then ask a *statistical query* which will be answered with respect to the specified distribution. In the PDC model the user can specify an arbitrary product distribution, and can ask for random examples with respect to that distribution. One could simulate the extended statistical query model in the PDC model by using random examples (drawn with respect to the specified distribution) to answer the statistical queries.

A PDC algorithm for finding relevant variables is implicit in the work of Bshouty and Feldman (2002). We discuss this algorithm in some detail in Section 10. Its running time is polynomial in n and its sample size, which is $O(n2^{16r}\log^2\frac{n}{\delta} + nr2^{16r}\log\frac{n}{\delta})$. It uses n distributions. Like our second new algorithm, when $r = O(\log n)$ it runs in time polynomial in n and $\log\frac{1}{\delta}$. Unlike our new algorithms, it chooses its distributions adaptively, and uses distributions that are not p-biased.

3. Yang (2001) gives an explicit lower bound for learning parity with honest statistical queries, and credits Jackson for proving this implicitly (Jackson, 2003). However, Jackson's proof is for a different statistical query learning model, and his proof does not work for honest statistical queries. Yang (2005) states a general lower bound that can be applied to parity. Its proof, in particular the discussion of "bad queries," seems to us to be incomplete.

Nevertheless, the distributions used by this algorithm are simple. In each, one parameter of the distribution is equal to 1/2, while the others are all equal to 1/4.

As noted in the introduction, it is possible to simulate membership queries in the PDC model by setting the parameters of the chosen product distribution to 0 and 1. The problem of efficiently learning Boolean functions with few relevant variables, using membership queries alone, has been addressed in a number of papers (Blum et al., 1995; Bshouty and Hellerstein, 1998; Damaschke, 2000). The goal in these papers is to have *attribute-efficient* algorithms that use a number of queries that is polynomial in r, the number of relevant variables, but only logarithmic in n, the total number of variables. Guijarro et al. (1999) investigated the problem of identifying relevant variables in the PAC model with membership queries.

In Section 10 we briefly describe a simple adaptive algorithm for identifying relevant variables using membership queries and uniform random examples. The algorithm is not novel; a similar approach is used in a number of algorithms for related problems (see, e.g., Arpe and Reischuk, 2007; Guijarro et al., 1999; Blum et al., 1995; Damaschke, 2000; Bshouty and Hellerstein, 1998). The algorithm runs in time polynomial in n and $\log\frac{1}{\delta}$, and uses $\log n + 1$ distinct product distributions. The time and sample complexity are lower for this algorithm than for the other PDC algorithms discussed in this paper, and for $r = \Omega(\log n)$, the number of product distributions used is lower as well. However, the other algorithms use only distributions whose parameters are bounded away from 0 and 1.

We use Fourier-based techniques in proving some of our results. There is an extensive literature on using Fourier methods in learning, including some of the papers mentioned above. Some of the most important results are described in the excellent survey of Mansour (1994).

Correlation immune functions and k-correlation immune functions have applications to secure communication, and have been widely studied in that field (see Roy, 2002, for a survey). Recent citations stem from the work of Siegenthaler (1984), but research on correlation immune functions predates those citations. Golomb (1999) has pointed out that his work in the 1950's on the classification of Boolean functions (Golomb, 1959) was motivated by the problem, useful for missile guidance, of designing bit sequences that would resist prediction methods based on correlation. During that period, as he states, such military applications "were not explicitly mentioned in the open literature."

Correlation immune functions have also been studied in other fields under different guises. The truth table of a k-correlation immune function corresponds to a certain orthogonal array (Camion et al., 1991). Orthogonal arrays are used in experimental design. The positive examples of a k-correlation immune function form a k-wise independent set. Such sets are used in derandomization (see, e.g., Alon, 1996).

It is natural to ask how many n-variable Boolean functions are correlation immune, since these actually *need* skewing. The question has been addressed in a number of different papers, as described by Roy (2002). Counts of correlation immune functions up to $n = 6$, separated by Hamming weight, were computed by Palmer et al. (1992). For larger n one can use the analytic approximation $2^{2^n} \cdot P_n$, where

$$P_n = \frac{1}{2}\left(\frac{8}{\pi}\right)^{n/2} 2^{-n^2/2}\left(1 - \frac{n^2}{4 \cdot 2^n}\right).$$

Since there are 2^{2^n} Boolean functions in all, P_n approximates the probability that a random Boolean function is correlation immune. Its main term was found by Denisov (1992), and the rest is the

beginning of an asymptotic series investigated by Bach (to appear). Even for small n, the above approximation is fairly accurate. For example, there are 503483766022188 6-variable correlation immune functions, and the above formula gives 4.99×10^{14}.

Skewing was developed as an applied method for learning correlation-immune Boolean functions. Skewing has also been applied to non-Boolean functions, and to Bayes nets (Lantz et al., 2007; Ray and Page, 2005).

The main results in Sections 8 and 9 of this paper appeared in preliminary form in Rosell et al. (2005).

6. Preliminaries

We begin with basic definitions and fundamental lemmas.

6.1 Notation and Terminology

We consider two-class learning problems, where the features, or variables, are Boolean. A *target function* is a Boolean function $f(x_1,\ldots,x_n)$. An *example* is an element of $\{0,1\}^n$. Example $a \in \{0,1\}^n$ is a *positive example* of Boolean function $f(x_1,\ldots,x_n)$ if $f(a)=1$, and a *negative example* of f if $f(a)=0$. A *labeled example* is an element $(a,b) \in \{0,1\}^n \times \{0,1\}$; it is a labeled example of f if $f(a)=b$.

Let $f(x_1,\ldots,x_n)$ be a Boolean function. The function f is a mapping from $\{0,1\}^n$ to $\{0,1\}$. An *assignment* $a=(a_1,\ldots,a_n)$ to the variables $x_1,\ldots,x_n$ is an element of $\{0,1\}^n$. The assignment obtained from a by negating the ith bit of a is denoted by $a_{\neg x_i}$. Given a Boolean function $f(x_1,\ldots,x_n)$, variable x_i is a *relevant variable* of f if there exists $a \in \{0,1\}^n$ such that $f(a) \neq f(a_{\neg x_i})$.

A *parity function* is a Boolean function $f(x_1,\ldots,x_n)$ such that for some $I \subseteq \{1,\ldots,n\}$, $f(x) = (\sum_{i \in I} x_i) \bmod 2$ for all $x \in \{0,1\}^n$.

For $\sigma \in \{0,1\}^n$, let $\sigma^i = (\sigma_1,\ldots,\sigma_{i-1},\sigma_{i+1},\ldots,\sigma_n)$, that is, σ^i denotes σ with its ith bit removed.

A *truth table* for a function f over a set of variables is a list of all assignments over the variables, together with the mapping of f for each assignment. For $i \in [1\ldots n]$ and $b \in \{0,1\}$, $f_{x_i \leftarrow b}$ denotes the function on $n-1$ variables produced by "hardwiring" the ith variable of f to b. More formally, $f_{x_i \leftarrow b} : \{0,1\}^{n-1} \rightarrow \{0,1\}$ such that for all $a \in \{0,1\}^{n-1}$, $f_{x_i \leftarrow b}(a) = f(a_1,a_2,\ldots,a_{i-1},b,a_i,\ldots,a_{n-1})$.

The integers between 1 and n are denoted by $[1\ldots n]$. For real a and b, (a,b) denotes the open interval from a to b.

For any probability distribution D, we use $\Pr_D$ and E_D to denote the probability and expectation with respect to distribution D. When D is defined on a finite set X and $A \subseteq X$, we define $\Pr_D(A)$ to be equal to $\sum_{a \in A} \Pr_D(a)$. We omit the subscript D when it is clear from context.

Given a probability distribution D on $\{0,1\}^n$, and a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, a *random example of f drawn with respect to D* is an example $(x, f(x))$ where x is drawn with respect to D.

A training set T for learning an n-variable Boolean function is a multiset consisting of elements in $\{0,1\}^n \times \{0,1\}$. It defines an associated distribution on $\{0,1\}^n \times \{0,1\}$ sometimes known as the *empirical distribution*. For each $(a,y) \in \{0,1\}^n \times \{0,1\}$, the probability of (a,y) under this distribution is defined to be the fraction of examples in the training set that are equal to (a,y). In the absence of noise, a training set for learning a function $f : \{0,1\}^n \rightarrow \{0,1\}$ is a set of labeled examples $(x, f(x))$. The empirical distribution on such a training set can be viewed as a distribution on $\{0,1\}^n$, rather than on $\{0,1\}^n \times \{0,1\}$.

A product distribution D on $\{0,1\}^n$ is a distribution defined by a parameter vector $[p_1,\ldots,p_n]$ in $[0,1]^n$ where for all $x \in \{0,1\}^n$, $\Pr_D[x] = (\prod_{i:x_i=1} p_i)(\prod_{i:x_i=0}(1-p_i))$. The uniform distribution on $\{0,1\}^n$ is the product distribution defined by $[1/2,1/2,\ldots,1/2]$. For fixed $p \in (0,1)$, we use $D[p]$ to denote the product distribution defined by $[p,\ldots,p]$. Distribution $D[p]$ is the *p-biased distribution*.

A *skew* is a pair (σ,p) where $\sigma \in \{0,1\}^n$ is an assignment, and $p \in (0,1)$. We refer to σ as the *orientation* of the skew, and p as the *weighting factor*.

Each skew (σ,p) induces a probability distribution $D_{(\sigma,p)}$ on the 2^n assignments in $\{0,1\}^n$ as follows. Let $\tau_p : \{0,1\} \times \{0,1\} \to \{p,1-p\}$ be such that for $b,b' \in \{0,1\}$, $\tau_p(b,b') = p$ if $b = b'$ and $\tau_p(b,b') = 1-p$ otherwise. For each $a \in \{0,1\}^n$, distribution $D_{(\sigma,p)}$ assigns probability $\Pi_{i=1}^n \tau_p(\sigma_i,a_i)$ to a. Thus distribution $D_{(\sigma,p)}$ is a product distribution in which every variable is set to 1 either with probability p, or with probability $1-p$. We call distributions $D_{(\sigma,p)}$ *skewed distributions*. When $\sigma = (1,\ldots,1)$, the distribution $D_{(\sigma,p)}$ is the p-biased distribution $D[p]$.

We note that in other papers on skewing, p is required to be in $(1/2,1)$, rather than in $(0,1)$. Here it is more convenient for us to let p be in $(0,1)$. Given any orientation σ, and any $p \in (0,1)$, skew $(\bar{\sigma},1-p)$, where $\bar{\sigma}$ is the bitwise complement of σ, induces the same distribution as (σ,p). Thus allowing p to be in $(0,1)$ does not change the class of skewed distributions, except that we also include the uniform distribution.

Given $a,b \in \{0,1\}^n$, let $\Delta(a,b) = |\{i \in [1,\ldots,n] | a_i \neq b_i\}|$, that is, $\Delta(a,b)$ is the Hamming distance between a and b. For $a,b \in \{0,1\}^n$, let $a+b$ denote the componentwise mod 2 sum of a and b. Given $c \in \{0,1\}^n$, we use $w(c)$ to denote the Hamming weight (number of 1's) of c. Thus $w(a+b) = \Delta(a,b)$.

In the *product distribution choice* (PDC) learning model, the learning algorithm has access to a special type of random example oracle for a target function $f(x_1,\ldots,x_n)$. This random example oracle takes as input the parameters $[p_1,\ldots,p_n]$ of a product distribution D over unlabeled examples $(x_1,\ldots,x_n)$. The oracle responds with a random example $(x_1,\ldots,x_n)$ drawn according to the requested distribution D, together with the value of the target f on that example. The learning algorithm is given as input a confidence parameter δ, where $0 < \delta < 1$. The algorithm is also given n as input.

6.2 Gain

Greedy tree learners partition a data set recursively, choosing a "split variable" at each step. They differ from one another primarily in their measures of "goodness" for split variables. The measure used in the well-known CART system is *Gini gain* (Breiman et al., 1984). Gini gain was also used in the decision tree learners employed in experimental work on skewing (Page and Ray, 2003; Ray and Page, 2004). In this paper, we use the term "gain" to denote Gini gain.

Gini gain is defined in terms of another quantity called the *Gini index*. Let S be a (multi) set of labeled examples. Let $S_1 = \{(x,y) \in S | y = 1\}$ and $S_0 = \{(x,y) \in S | y = 0\}$. The Gini index of S is $2\frac{|S_1||S_0|}{|S|^2}$. Let $\tilde{H}(S)$ denote the Gini index of S.

Let x_i be a potential split variable. Let $T_1 = \{(x,y) \in S | x_i = 1\}$ and $T_0 = \{(x,y) \in S | x_i = 0\}$. The Gini index of S *conditional on* x_i is defined to be $\tilde{H}(S|x_i) := \frac{|T_1|}{|S|}\tilde{H}(T_1) + \frac{|T_0|}{|S|}\tilde{H}(T_0)$. In decision tree terms, this is the weighted sum of the Gini indices of the child nodes resulting from a split on x_i. The *Gini gain* of x_i with respect to S is

$$G(S,x_i) = \tilde{H}(S) - \tilde{H}(S|x_i).$$

The Gini gain is always a value in the interval $[0,1/2]$. Some definitions of Gini gain and Gini index differ from the one above by a factor of 2; our definition follows that of Breiman et al. (1984).

Now suppose that each example in our (multi) set S has an associated *weight*, a real number between 0 and 1. We can define the gain on this weighted set by modifying the above definitions in the natural way: each time the definitions involve the size of a set, we instead use the sum of the weights of the elements in the set.

We can also define Gini index and Gini gain of variable x_i with respect to $f : \{0,1\}^n \to \{0,1\}$ under a distribution D on $\{0,1\}^n$. The Gini index of f with respect to a probability distribution D on $\{0,1\}^n$ is $2\Pr_D[f=1](1-\Pr_D[f=1])$. Let $\tilde{H}_D(f)$ denote the Gini index of f with respect to D. For any potential split variable x_i, the Gini index of f with respect to D, *conditional on* x_i is $\tilde{H}_D(f|x_i) := \Pr_D[x_i=0]\tilde{H}_D(f_{x_i \leftarrow 0}) + \Pr_D[x_i=1]\tilde{H}_D(f_{x_i \leftarrow 1})$. The *Gini gain* of a variable x_i with respect to f, under distribution D, is

$$G_D(f,x_i) = \tilde{H}_D(f) - \tilde{H}_D(f|x_i).$$

The Gini gain of x_i with respect to f, under the uniform distribution on $\{0,1\}^n$, is equal to the Gini gain of x_i with respect to the training set T consisting of all entries in the truth table of f.

Given a skew (σ, p) and a function f, the Gini gain of a variable x_i with respect to f under distribution $D_{(\sigma,p)}$ is equivalent to the gain that is calculated, using the procedure described in Section 4, by applying skew (σ, p) to the training set T consisting of the entire truth table for f.

The following lemma relates the size of the Gini gain with respect to a distribution D to the difference in the conditional probabilities $\Pr_D[f=1|x_i=1] - \Pr_D[f=1|x_i=0]$.

Lemma 1 *Let f be an n-variable Boolean function, and let D be a distribution on $\{0,1\}^n$ such that $\Pr[x_i=1]$ is strictly between 0 and 1. Then $G_D(f,x_i)$, the Gini gain of variable x_i with respect to f, under distribution D, is equal to*

$$2p_i(1-p_i)(Pr_D[f=1|x_i=1] - Pr_D[f=1|x_i=0])^2$$

where $p_i = \Pr_D[x_i=1]$.

Proof. Let $p = p_i$, $\beta = \Pr_D[f=1]$, $\beta_1 = \Pr_D[f=1|x_i=1]$, and $\beta_0 = \Pr_D[f=1|x_i=0]$. Thus $\beta = p\beta_1 + (1-p)\beta_0$.

The Gini gain of x_i with respect to f is

$$\begin{aligned}
&2(\beta(1-\beta) - p(\beta_1(1-\beta_1)) - (1-p)(\beta_0(1-\beta_0))) \\
&\quad = 2(\beta(1-\beta) - (p\beta_1 + (1-p)\beta_0)) + p\beta_1^2 + \beta_0^2(1-p) \\
&\quad = 2(\beta(1-\beta) - \beta + p\beta_1^2 + \beta_0^2(1-p)) \\
&\quad = 2(-\beta^2 + p\beta_1^2 + \beta_0^2(1-p)).
\end{aligned}$$

Substituting $p\beta_1 + (1-p)\beta_0$ for β, we get that the last quantity is

$$\begin{aligned}
&= 2(-p^2\beta_1^2 - 2p(1-p)\beta_0\beta_1 - (1-p)^2\beta_0^2 + p\beta_1^2 + \beta_0^2(1-p)) \\
&= 2((1-p)p(\beta_1^2 - 2\beta_0\beta_1 + \beta_0^2)) \\
&= 2p(1-p)(\beta_1 - \beta_0)^2
\end{aligned}$$

□

Under distribution D on $\{0,1\}^n$, x_i and (the output of) f are independent iff $G_D(f,x_i)=0$.

6.3 Fourier Coefficients

Given a Boolean function $f:\{0,1\}^n\to\{0,1\}$, define an associated function $F=1-2f$. That is, $F:\{0,1\}^n\to\{1,-1\}$ is such that $F(x)=1-2f(x)$ for all $x\in\{0,1\}^n$. The function F can be seen as an alternative representation of Boolean function f, using -1 and 1 respectively to represent true and false outputs, rather than 1 and 0.

For every $z\in\{0,1\}^n$, let $\chi_z:\{0,1\}^n\to\{1,-1\}$ be such that $\chi_z(x)=(-1)^{\sum_{i=1}^n x_i z_i}$. Thus χ_z is the alternative representation of the function computing the parity of the variables set to 1 by z. For $z\in\{0,1\}^n$, n-variable Boolean function f, and associated $F=1-2f$, the *Fourier coefficient* $\hat{f}(z)$ is

$$\hat{f}(z):=E[F(x)\chi_z(x)]$$

where the expectation is with respect to the uniform distribution on $x\in\{0,1\}^n$.

The *degree* of Fourier coefficient $\hat{f}(z)$ is $w(z)$, the Hamming weight of z. The Fourier coefficient *associated with the variable* x_i is $\hat{f}(z)$ where z is the characteristic vector of x_i (i.e., $z_i=1$ and for $j\neq i$, $z_j=0$). In an abuse of notation, we will use $\hat{f}(x_i)$ to denote this Fourier coefficient. Thus $\hat{f}(x_i)=E[F(x)(1-2x_i)]$. The function F can be expressed by its Fourier series, as we have $F(x)=\sum_{z\in\{0,1\}^n}\hat{f}(z)\chi_z(x)$.

Fourier coefficients can be generalized from the uniform distribution to product distributions, as described by Furst et al. (1991). Let D be a product distribution on $\{0,1\}^n$ defined by parameters $[p_1,\ldots,p_n]$, all of which are strictly between 0 and 1. For $z\in\{0,1\}^n$, let $\phi_{D,z}:\{0,1\}^n\to\{0,1\}$ be such that $\phi_{D,z}(x)=\prod_{i:z_i=1}\frac{\mu_i-x_i}{\sigma_i}$ where $\mu_i=p_i$ is $E_D[x_i]$ and $\sigma_i=\sqrt{p_i(1-p_i)}$ is the standard deviation of x_i under D. The Fourier coefficient $\hat{f}_D(z)$, for product distribution D, is

$$\hat{f}_D(z):=E_D[F(x)\phi_{D,z}(x)].$$

When D is the uniform distribution, this is the ordinary Fourier coefficient.

Parseval's identity, applied to the Fourier coefficients of product distributions, states that

$$\sum_{z\in\{0,1\}^n}\hat{f_D}^2(z)=1.$$

The Fourier coefficient associated with the variable x_i, with respect to product distribution D, is $\hat{f}_D(z)$, where z is the characteristic vector of x_i. Abusing notation as before, we will use $\hat{f}_D(x_i)$ to denote this Fourier coefficient. Thus

$$\hat{f}_D(x_i)=\frac{p_iE_D[F(x)]-E_D[x_iF(x)]}{\sqrt{p_i(1-p_i)}}.$$

The next lemma shows that the gain of a variable and its Fourier coefficient are closely related.

Lemma 2 *Let f be an n-variable Boolean function, and let D be a product distribution over $\{0,1\}^n$ defined by $[p_1,\ldots,p_n]$, such that each $p_i\in(0,1)$. Then*

$$\hat{f}_D(x_i)=2\sqrt{p_i(1-p_i)}(\Pr_D[f=1|x_i=1]-\Pr_D[f=1|x_i=0])$$

and

$$G_D(f,x_i)=\hat{f}_D^2(x_i)/2.$$

Proof. By definition,

$$\hat{f}(x_i) = \frac{p_i E_D[F(x)] - E_D[x_i F(x)]}{\sqrt{p_i(1-p_i)}}.$$

Let $\beta = \Pr_D[f = 1]$ (which equals $\Pr_D[F = -1]$), $\beta_1 = \Pr_D[f = 1 | x_i = 1]$, and $\beta_0 = \Pr_D[f = 1 | x_i = 0]$.

Since $p_i E_D[F(x)] = p_i(1-2\beta)$, $E_D[F(x)x_i] = p_i(1-2\beta_1)$, and $\beta = p_i\beta_1 + (1-p_i)\beta_0$, it follows that

$$\begin{aligned} p_i E_D[F(x)] - E_D[x_i F(x)] &= 2p_i(-\beta + \beta_1) \\ &= 2p_i(-p_i\beta_1 - (1-p_i)\beta_0 + \beta_1) \\ &= 2p_i(1-p_i)(\beta_1 - \beta_0). \end{aligned}$$

Dividing by $\sqrt{p_i(1-p_i)}$, we have that

$$\hat{f}_D(x_i) = 2\sqrt{p_i(1-p_i)}(\Pr_D[f = 1 | x_i = 1] - \Pr_D[f = 1 | x_i = 0]).$$

The lemma follows immediately from Lemma 1. □

The following important facts about first-order Fourier coefficients for product distributions are easily shown. For D a product distribution on $\{0,1\}^n$ where each $p_i \in (0,1)$,

1. If x_i is an irrelevant variable of a Boolean function f, then $\hat{f}_D(x_i) = 0$.

2. $G_D(f, x_i) = 0$ iff $\hat{f}_D(x_i) = 0$.

6.4 Correlation Immune Functions

For $k \geq 1$, a Boolean function is defined to be *k-correlation immune* if for all $1 \leq d \leq k$, all degree-d Fourier coefficients of f are equal to 0. An equivalent definition is as follows (Xiao and Massey, 1988; Brynielsson, 1989). Let $x_1, \ldots, x_n$ be random Boolean variables, each chosen uniformly and independently. Let $y = f(x_1, \ldots, x_n)$. Then f is k-correlation immune if and only if, for any distinct variables $x_{i_1}, \ldots, x_{i_k}$ of f, the variables $y, x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ are mutually independent.

A greedy decision tree learner would have difficulty learning k-correlation immune functions using only k-lookahead; to find relevant variables in the presence of irrelevant ones for such functions, it would need to use $k+1$-lookahead.

A Boolean function is *correlation immune* if it is 1-correlation immune. Equivalently, a Boolean function f is correlation immune if all variables of f have zero gain for f, with respect to the uniform distribution on $\{0,1\}^n$. As can be seen from Lemma 1, this is the case iff for every input variable x_i of the function, $\Pr[f = 1 | x_i = 1] = \Pr[f = 1 | x_i = 0]$, where probabilities are with respect to the uniform distribution on $\{0,1\}^n$. The following alternative characterization of correlation-immune functions immediately follows: A Boolean function is correlation-immune iff

$$|\{a \in \{0,1\}^n \mid f(a) = 1 \text{ and } a_i = 1\}| = |\{a \in \{0,1\}^n \mid f(a) = 1 \text{ and } a_i = 0\}|.$$

6.5 Correlation Immune Functions for Product Distributions

Correlation immune functions are defined with respect to the uniform distribution. Here we extend the definition to apply to arbitrary product distributions with parameters strictly between 0 and 1. In particular, for such a product distribution D, we can define a function to be *correlation immune for* D if either (1) The degree-1 Fourier coefficients with respect to D are all 0, or (2) the gain of every variable with respect to D is 0, or (3) $\Pr_D[f = 1|x_i = 1] - \Pr_D[f = 1|x_i = 0] = 0$ for all variables x_i of f. By the results in Section 6, these conditions are equivalent.[4]

A natural question is whether there are (non-constant) correlation immune functions for non-uniform product distributions D. There are, as illustrated by the following example, which can be easily generalized to other similar product distributions.

6.5.1 EXAMPLE

Let n be a multiple of 3, and let D be the product distribution defined by $[2/3, 2/3, \ldots, 2/3]$.

For any n that is a multiple of 3, we will show that the following function f is correlation immune with respect to D.

Let f be the n-variable Boolean function such that $f(x) = 1$ if $x = 110110110110\ldots$ (i.e., $n/3$ repetitions of 110), or when x is equal to one of the two right-shifts of that vector. For all other x, $f(x) = 0$.

To prove correlation immunity, it suffices to show that for each x_i, $\Pr_D[f = 1|x_i = 1] = \Pr_D[f = 1]$.

Each positive example of f has the same probability. It is easy to verify that for each x_i, 2/3 of the positive examples have $x_i = 1$. Thus $\Pr_D[f = 1 \text{ and } x = 1] = 2/3 \Pr_D[f = 1]$. So,

$$\begin{aligned} \Pr_D[f = 1|x = 1] &= \Pr_D[f = 1 \text{ and } x = 1]/\Pr_D[x = 1] \\ &= (2/3\Pr_D[f = 1])/(2/3) \\ &= \Pr_D[f = 1] \end{aligned}$$

□

In Section 9 we will give examples of product distributions D for which there are no correlation-immune functions.

7. Estimating First-order Fourier Coefficients and Gain

Fourier-based learning algorithms work by computing estimates of selected Fourier coefficients using a sample. Given a training set $S = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ for a Boolean function f and $z \in \{0,1\}^n$, the *estimated Fourier coefficient for z, calculated on S, with respect to product distribution D,* is

$$\hat{f}_{S,D}(z) := \frac{1}{m} \sum_{j=1}^{m} (1 - 2y^{(j)}) \phi_{D,z}(x^{(j)}).$$

We will use $\hat{f}_{S,D}(x_i)$ to denote $\hat{f}_{S,D}(z)$, where z is the characteristic vector of x_i.

4. We do not extend the definition of correlation-immunity to non-product distributions. With respect to a non-product distribution, it is possible for both relevant and irrelevant variables to have non-zero gain.

To simplify notation, where D is clear from context, we will often write $\hat{f}_S(z)$ instead of $\hat{f}_{S,D}(z)$. Since $\phi_{D,z}$ depends on D, calculating $\hat{f}_S(z)$ from S requires knowledge of D. Since we will apply this lemma in the context of the PDC model, in which D is known, this is not a problem for us.

If S is a random sample of f drawn with respect to D, then $\hat{f}_D(z) = E_D[(1-2f(x))\phi_{D,z}(x)]$ and $\hat{f}_{S,D}(z)$ is the estimate of the expectation $E_D[(1-2f(x))\phi_{D,z}(x)]$ on sample S.

In Section 10, there are situations in which we will know that, with respect to a known product distribution D, there exists a relevant variable of a function f whose first-order Fourier coefficient has magnitude at least q, for some value q. As mentioned earlier, the first-order Fourier coefficients of irrelevant variables are zero. Thus if one can estimate first-order Fourier coefficients of f so the estimates each have additive error less than $q/2$, then a non-empty subset of the relevant variables of f can be constructed by taking all variables whose Fourier coefficient estimates are at least $q/2$. The following lemma gives an upper bound on the sample size that would be needed to produce the desired estimates with high probability (by setting $\varepsilon = q/2$). The lemma is implicit in the paper of Furst et al. (1991), and follows from a standard bound of Hoeffding.

Lemma 3 *Let f be an n-variable Boolean function and let D be a product distribution over $\{0,1\}^n$ defined by $[p_1,\ldots,p_n]$. Let $\beta = \max_i\{1/p_i, 1/(1-p_i)\}$, $\varepsilon > 0$, and $0 < \delta < 1$. If S is a set of at least*

$$\frac{1}{\varepsilon^2} 2(\beta-1)\ln\frac{2n}{\delta}$$

random examples of f, drawn from distribution D, then with probability at least $1-\delta$, $|\hat{f}_{S,D}(x_i) - \hat{f}_D(x_i)| < \varepsilon$ for all variables x_i of f.

The above lemma is useful only in situations in which the parameters of D are known, so that $\hat{f}_D$ can be computed. A similar bound can be applied when D is an unknown product distribution, and its parameters are estimated from the sample (see Furst et al., 1991).

Skewing works by estimating gain, rather than by estimating first-order Fourier coefficients. More generally, one can use gain estimates rather than Fourier coefficient estimates to try to identify relevant variables of a function (assuming some have non-zero gain). Below in Lemma 6 we give a sample-complexity bound for estimating gain. We prove this bound using martingales. In contrast to the bound given in Lemma 3, this bound can be applied in cases where the distribution is unknown and arbitrary (i.e., it does not have to be a product distribution).

Before presenting the martingale-based bound, however, we first prove a bound that easily follows from the work of Furst et al. (1991) and the relationship between gain and first-order Fourier coefficients given in Lemma 2. The bound itself is the same as the bound for estimating Fourier coefficients given in Lemma 3. Algorithmically, the bound applies to the following procedure for estimating $G(D,x_i)$, when D is a known product distribution. Given a sample S, use it to compute the estimate $\hat{f}_S(x_i)$ of the Fourier coefficient of x_i. If $\hat{f}_S(x_i)$ is in the interval $[-1,1]$, then let $\tilde{f}_S(x_i) = \hat{f}_S(x_i)$, otherwise, let $\tilde{f}_S(x_i) = 1$ if $\hat{f}_S(x_i)$ is positive, and $\tilde{f}_S(x_i) = -1$ otherwise. Thus $\tilde{f}_S(x_i)$ is $\hat{f}_S(x_i)$, restricted to the interval [-1,1]. Output $(\tilde{f}_S(x_i))^2/2$ as the estimate for $G_D(f,x_i)$.

Lemma 4 *Let f be an n-variable Boolean function and let D be a product distribution over $\{0,1\}^n$ defined by $[p_1,\ldots,p_n]$. Let $\beta = \max_i\{1/p_i, 1/(1-p_i)\}$, $\varepsilon > 0$, and $0 < \delta < 1$. If S is a set of*

$$\frac{1}{\varepsilon^2} 2(\beta-1)\ln\frac{2n}{\delta}$$

random examples of f, drawn from distribution D, then with probability at least $1-\delta$, $|(\tilde{f}_S(x_i))^2/2 - G_D(f,x_i)| \leq \varepsilon$.

Proof. By Lemma 2, $G_D(f,x_i) = \frac{\hat{f}_D^2(x_i)}{2}$. Let $Y = \hat{f}_D(x_i)$ and let $\tilde{Y} = \tilde{f}_S(x_i)$. By Lemma 3, with probability at least $1-\delta$, $|\hat{f}_S(x_i) - Y| < \varepsilon$. As noted by Furst et al. (1991), since Y is a Fourier coefficient, $Y \in [-1,1]$, and thus restricting the estimate of Y to $[-1,1]$ can only increase its accuracy. Thus $|\tilde{Y} - Y| < \varepsilon$ as well. It follows that $|\tilde{Y}^2/2 - G_D(f,x_i)| = |\tilde{Y}^2/2 - Y^2/2| = \frac{1}{2}|(\tilde{Y}-Y)(\tilde{Y}+Y)| \leq \varepsilon$, since $|\tilde{Y}+Y| \leq 2$. □

The bound in the above lemma is similar to the martingale-based bound we give below in Lemma 6. The main difference is that it has a factor of $(\beta - 1)$, meaning that it depends on p_i. In Section 10, Theorem 10.2, we apply Lemma 6 to prove a sample complexity result for an algorithm in the PDC model. In this context, p_i is not constant, and applying the bound in Lemma 4 instead would yield a slightly worse sample complexity for the algorithm (by a factor of $O(r)$). We now proceed with the presentation of the martingale-based bound. The bound is based on a standard large deviation estimate, which can be thought of as a "vector" version of the Chernoff bound. It implies that a martingale is unlikely to wander too far from its initial value.

We recall some definitions. Let $Z(0), Z(1), \ldots$ be a discrete-time Markov process in $\mathbb{R}^k$ with differences bounded by c. That is, $Z(0), Z(1), \ldots$ are random variables taking values in $\mathbb{R}^k$, such that the distribution of $Z(t+1)$ given $Z(u)$ for all $u \leq t$ depends only on $Z(t)$, and for each pair $Z(t), Z(t+1)$ the L_2 norm $||Z(t+1) - Z(t)||$ is at most c. We call the process a *martingale* if for all $t \geq 0$, $E[Z(t)]$ exists, and $E[Z(t+1)|Z(t)] = Z(t)$. (More general definitions exist, but this will suffice for our purpose.)

Lemma 5 *Let $Z(t)$ be a martingale in R^k with differences bounded by c. Then for any $\lambda > 0$,*

$$\Pr[\ ||Z(t) - Z(0)|| \geq \lambda\ ] \leq 2\exp(\frac{-\lambda^2}{2tc^2}). \tag{1}$$

Proof See, for example, Pinelis (1992). □

Lemma 6 *Let f be an n-variable Boolean function and let D be a product distribution over $\{0,1\}^n$ whose parameters are in $(0,1)$. Let $\varepsilon > 0$, and $0 < \delta < 1$. If S is a set of at least*

$$256\ln(2n/\delta)/\varepsilon^2$$

random examples of f, drawn from distribution D, then with probability at least $1-\delta$, $|G(S,x_i) - G_D(f,x_i)| \leq \varepsilon$ for all variables x_i of f.

Proof Let x_i be a variable, and consider the 2×2 table

	$f=0$	$f=1$
$x_i = 0$	a_1	a_2
$x_i = 1$	a_3	a_4

In this table, the a_j's are probabilities, so that a_1 denotes the probability (under D) that $x_i = f = 0$, and similarly for the others. Therefore, $0 \leq a_j \leq 1$, and $\sum a_j = 1$.

By drawing a random sample S of f from distribution D, we get counts m_1, m_2, m_3, m_4 corresponding to the a_j's. For example, m_2 is the number of examples in S for which $x_i = 0$ and $f = 1$. We can view the sampling procedure as happening over time, where the tth example is drawn at time t.

At times $t = 0, 1, 2, \ldots$, we can observe

$$Z(t) := (m_1 - a_1 t, m_2 - a_2 t, m_3 - a_3 t, m_4 - a_4 t).$$

By the definition of Z,

$$\begin{aligned} E[Z(t+1) - Z(t) | Z(t)] &= a_1(1-a_1, -a_2, -a_3, -a_4) + a_2(-a_1, 1-a_2, -a_3, -a_4) \\ &\quad + a_3(-a_1, -a_2, 1-a_3, -a_4) + a_4(-a_1, -a_2, -a_3, 1-a_4) \\ &= (0,0,0,0) \end{aligned}$$

where the last equation follows because $\sum a_j = 1$. Thus $Z(0), Z(1), \ldots$ is a martingale in $\mathbf{R}^4$. Also, $Z(t+1) - Z(t)$ equals, up to symmetry, $(1-a_1, -a_2, -a_3, -a_4)$. Since $a_2^2 + a_3^2 + a_4^2 \leq 1$,

$$(1-a_1)^2 + a_2^2 + a_3^2 + a_4^2 \leq 2,$$

and the martingale has differences bounded by $c = \sqrt{2}$.

The gain of x_i in f with respect to distribution D is

$$G_D(f, x_i) = 2\left[\beta(1-\beta) - p\beta_1(1-\beta_1) - (1-p)\beta_0(1-\beta_0)\right]$$

where

$$\beta = \Pr[f = 1] = a_2 + a_4,$$

$$p = \Pr[x_i = 1] = a_3 + a_4,$$

$$\beta_0 = \Pr[f = 1 | x_i = 0] = \frac{a_2}{a_1 + a_2},$$

and

$$\beta_1 = \Pr[f = 1 | x_i = 1] = \frac{a_4}{a_3 + a_4}.$$

Substituting these into the above gain formula and simplifying, we get

$$G_D(f, x_i) = 2\left[(a_1 + a_3)(a_2 + a_4) - \frac{a_3 a_4}{a_3 + a_4} - \frac{a_1 a_2}{a_1 + a_2}\right].$$

Define the function $G(a_1, \ldots, a_4)$ to be equal to the right hand side of the above equation. This is a continuous function of the a_j's, on the simplex $a_j \geq 0, \sum a_j = 1$.

Observe that

$$0 < \frac{\partial}{\partial a_j}\left(\frac{a_j a_k}{a_j + a_k}\right) = \frac{1}{(a_j/a_k + 1)^2} < 1,$$

if $a_j, a_k > 0$, and

$$0 \leq \frac{\partial}{\partial a_j}(a_1 + a_3)(a_2 + a_4) \leq \sum a_i = 1.$$

This implies that $\left|\partial G / \partial a_j\right| \leq 2$ in the interior of the simplex.

Suppose that $b = (b_1, b_2, b_3, b_4)$ and $c = (c_1, c_2, c_3, c_4)$ are two points on the interior of the simplex with $\max_j\{|c_j - b_j|\} = \mu$. Let $a(t) = b + t(c - b)$ be the the parametric equation of the line from b to c, and let $\tilde{G}(t) = G(a(t))$.

Letting $a_i(t)$ be the ith coordinate of $a(t)$, and applying the chain rule, we get that

$$\frac{\partial \tilde{G}}{\partial t} = \sum_i \frac{\partial \tilde{G}}{\partial a_i} \frac{da_i}{dt}. \tag{2}$$

Since $\tilde{G}(0) = G(b)$ and $\tilde{G}(1) = G(c)$, by the mean value theorem, there exists $t^* \in [0,1]$ such that

$$\frac{\partial \tilde{G}}{\partial t}(t^*) = G(c) - G(b). \tag{3}$$

For $(a_1, \ldots, a_4)$ in the interior of the simplex, $\left|\partial \tilde{G}/\partial a_i\right| \leq 2$. By the definition of $a(t)$, $|da_i/dt| = |c_i - b_i| \leq \mu$. Thus (2) and (3) imply that

$$|G(c) - G(b)| \leq 8\mu. \tag{4}$$

Since G is continuous, this holds even for probability vectors b and c on the boundary.

We seek a sample size m large enough that (for all variables x_i)

$$\Pr[\ |G(S, x_i) - G_D(f, x_i)| \geq \varepsilon\ ] \leq \frac{\delta}{n}.$$

Let the empirical frequencies be $\hat{a}_j = m_i/m$, $i = 1, \ldots, 4$. By (4), it will suffice to make m large enough that, with probability at least $1 - \delta/n$, we observe $|\hat{a}_j - a_j| < \varepsilon/8$ for all j. Let's call a sample "bad" if for some j, $|m_j/m - a_j| \geq \varepsilon/8$. Since $Z(0) = \vec{0}$, this implies that $||Z(m) - Z(0)|| \geq \varepsilon m/8$. If we take $\lambda = \varepsilon m/8$, $c = \sqrt{2}$, and $t = m$ in the Chernoff bound (1), we see that

$$\Pr[\text{ bad sample }] \leq 2e^{-\frac{\varepsilon^2 m}{256}}.$$

This will be less than δ/n as soon as

$$m \geq \frac{256 \ln(2n/\delta)}{\varepsilon^2}.$$

□

8. A Property of Non-constant Boolean Functions

In this section we prove a property of Boolean functions that we will use repeatedly in subsequent sections. The property is given in the following lemma.

For $k \in [0, \ldots, n]$, let $W_k(f)$ denote the number of positive assignments of f of Hamming weight k.

Lemma 7 *Let f be a non-constant Boolean function on $\{0,1\}^n$. Then there exists a variable x_i of f and a number $k \in [0, \ldots, n-1]$ such that $W_k(f_{x_i \leftarrow 0}) \neq W_k(f_{x_i \leftarrow 1})$.*

Proof. Assume no such variable x_i exists.

Without loss of generality, assume that $f(0^n) = 1$. We prove that for all $a \in \{0,1\}^n$, $f(a) = 1$. The proof is by induction on the Hamming weight of a, $w(a)$. The base case clearly holds.

Now let $j \in [0,\ldots,n-1]$. Assume inductively that all assignments x of Hamming weight j satisfy $f(x) = 1$. Let $l \in [1,\ldots,n]$. Let $t \in \{0,1\}^n$ be an arbitrary assignment of Hamming weight j such that $t_l = 0$; t exists because $j < n$. By the initial assumption, $W_j(f_{x_l \leftarrow 0}) = W_j(f_{x_l \leftarrow 1})$. Further, by the inductive assumption, for every assignment u such that $w(u) = j$, $f(u) = 1$. There are precisely $\binom{n-1}{j}$ assignments u such that $w(u) = j$ and $u_l = 0$. All these assignments u satisfy $f(u) = 1$, and thus $W_j(f_{x_l \leftarrow 0}) = \binom{n-1}{j}$. Therefore $W_j(f_{x_l \leftarrow 1}) = \binom{n-1}{j}$ also. The quantity $\binom{n-1}{j}$ is equal to the total number of assignments in $\{0,1\}^{n-1}$ of Hamming weight j. It follows that $f_{x_l \leftarrow 1}(b) = 1$ for all $b \in \{0,1\}^{n-1}$ of Hamming weight j, and hence $f(a) = 1$ for all $a \in \{0,1\}^n$ such $a_l = 1$ and $w(a) = j+1$. Since index l is arbitrary, and each assignment of Hamming weight $j+1$ has at least one variable set to 1, it follows that $f(a) = 1$ for all $a \in \{0,1\}^n$ of Hamming weight $j+1$.

We have thus shown by induction that $f(a) = 1$ for all $a \in \{0,1\}^n$. This contradicts the property that f is a non-constant function. □

Lemma 7 can be restated using the terminology of *weight enumerators*. Given a binary code (i.e., a subset C of $\{0,1\}^n$, for some n), the weight enumerator of this code is the polynomial $P(z) = \sum_k W_k z^k$, where W_k is the number of codewords (elements of C) of Hamming weight k. Lemma 7 states that if f is a non-constant Boolean function, then it has a relevant variable x_i such that codes $C_0 := \{x \in \{0,1\}^{n-1} | f_{x_i \leftarrow 0}(x) = 1\}$, and $C_1 := \{x \in \{0,1\}^{n-1} | f_{x_i \leftarrow 1}(x) = 1\}$ have different weight enumerators.

Lemma 7 proves the existence of a variable x_i with a given property. One might conjecture that all relevant variables of f would share this property, but this is not the case, as shown in the following simple example.

8.1 Example

Let $f(x_1,x_2,x_3) = (\neg x_1 \vee \neg x_2 \vee x_3)(x_1 \vee x_2 \vee \neg x_3)$. Let $\sigma = (0,0,0)$. Since $f(1,1,0) \neq f(0,1,0)$, x_1 is a relevant variable of f. It is straightforward to verify that, for $k \in \{0,1,2\}$, $W_k(f_{x_1 \leftarrow 0}) = W_k(f_{x_1 \leftarrow 1})$. The same holds for x_2 by symmetry. Variable x_3 is the only one satisfying the property of Lemma 7.

9. Skewing Given the Entire Truth Table

In this section, we analyze skewing in an idealized setting, where the available data consists of the full truth table of a Boolean function. We then do an analysis of sequential skewing in the same setting.

9.1 A Motivating Example

Recall that a correlation immune function $f(x_1,\ldots,x_n)$ is one such that for every variable x_i, the gain of x_i with respect to f is 0 under the uniform distribution on $\{0,1\}^n$. We are interested in the following question: When skewing is applied to a correlation immune function, will it cause a relevant variable to have non-zero gain under the skewed distribution? (Equivalently, will it cause one of the first-order Fourier coefficients to become non-zero?) We show that, in the idealized

setting, the answer to this question is "yes" for nearly all skews. The answer is somewhat different for sequential skewing.

When we use a skew (σ, p) to reweight a data set that consists of an entire truth table, the weight assigned to each assignment a in the truth table by the skewing procedure is $P_{D(\sigma,p)}(a)$, where $D(\sigma, p)$ is the skewed distribution defined by (σ, p). Moreover, the gain of a variable x_i as measured on the weighted truth table is precisely the gain with respect to $D(\sigma, p)$. By Lemma 1, it follows that a variable x_i will have gain on the skewed (weighted) truth table data set iff $P_D(f = 1|x_i = 1) - P_D(f = 1|x_i = 0) \neq 0$, where $D = D(\sigma, p)$. If x_i is a relevant variable, the difference $P_D(f = 1|x_i = 1) - P_D(f = 1|x_i = 0)$ can be expressed as a polynomial $h(p)$ in p of degree at most $r-1$, where r is the number of relevant variables of f. If x_i is an irrelevant variable, $P_D(f = 1|x_i = 1) - P_D(f = 1|x_i = 0) = 0$. The main work in this section will be to show that for some relevant variable x_i, this polynomial is not identically 0. Having proved that, we will know that for at most $r-1$ values of weight factor p (the roots of h), $h(p) = 0$. For all other values of p, $h(p) \neq 0$, and x_i has gain in f with respect to $D(\sigma, p)$.

We give an example construction of the polynomial $h(p)$ for a particular function and skew. Consider a Boolean function f over 5 variables whose positive examples are $(0,0,0,1,0)$, $(0,0,1,0,0)$, $(1,0,1,1,0)$. Assume a skew (σ, p) where $\sigma = (1,\ldots,1)$ and p is some arbitrary value in $(0,1)$. Let $D = D_{(\sigma,p)}$. There are two positive examples of f setting $x_1 = 0$, namely $(0,0,0,1,0)$ and $(0,0,1,0,0)$. It is easy to verify that $P_D(f = 1|x_1 = 0) = 2p(1-p)^3$. Similarly, $P_D(f = 1|x_1 = 1) = p^2(1-p)^2$. Let $h(p) = P_D(f = 1|x_1 = 1) - P_D(f = 1|x_1 = 0)$. Then $h(p) = p^2(1-p)^2 - 2p(1-p)^3$, which is a degree-4 polynomial in p. This polynomial has at most 4 roots, and it is not identically 0. It follows that for all but at most 4 choices of p, $h(p)$ is not zero. Thus if we choose p uniformly at random from $(0,1)$, with probability 1, x_1 has gain for (f, σ, p).

9.2 Analysis of Skewing Given the Complete Truth Table

For $f : \{0,1\}^n \to \{0,1\}$ a Boolean function, $k \in [1\ldots n]$, and $\sigma \in \{0,1\}^n$, let $W(f, \sigma, k)$ denote the number of assignments $b \in \{0,1\}^n$ such that $f(b) = 1$ and $\Delta(b, \sigma) = k$.

Using the symmetry of the Boolean hypercube, we can generalize Lemma 7 to obtain the following lemma, which we will use in our analysis of skewing.

Lemma 8 *Let f be a non-constant Boolean function on $\{0,1\}^n$, $\sigma \in \{0,1\}^n$ be an orientation, and $i \in [1\ldots n]$. Then there exists a variable x_i of f and $k \in [0,\ldots,n-1]$ such that $W(f_{x_i \leftarrow 1}, \sigma^i, k) \neq W(f_{x_i \leftarrow 0}, \sigma^i, k)$.*

Proof. Recall that given two assignments a and b, we use $a+b$ to denote componentwise addition mod 2. Let $f' : \{0,1\}^n \to \{0,1\}$ be such that $f'(x) = f(x+\sigma)$.

Applying Lemma 7 to function f', let x_i and k be such that $W_k(f'_{x_i \leftarrow 1}) \neq W_k(f'_{x_i \leftarrow 0})$.

For all $a \in \{0,1\}^{n-1}$, $f'_{x_i \leftarrow \sigma_i}(a) = 1$ and $w(a) = k$ iff $f_{x_i \leftarrow 0}(a + \sigma^i) = 1$ and $\Delta(a + \sigma^i, \sigma^i) = w((a + \sigma^i) + \sigma^i) = k$. It follows that $W_k(f'_{x_i \leftarrow \sigma_i}) = W(f_{x_i \leftarrow 0}, \sigma^i, k)$. The analogous statement holds for $W_k(f'_{x_i \leftarrow \neg\sigma_i})$. Thus $W(f_{x_i \leftarrow 1}, \sigma^i, k) \neq W(f_{x_i \leftarrow 0}, \sigma^i, k)$. □

We now show the connection between the above lemma and gain.

Lemma 9 *Let f be a Boolean function on $\{0,1\}^n$, $\sigma \in \{0,1\}^n$ be an orientation, and $i \in [1\ldots n]$. Let r be the number of relevant variables of f. If $W(f_{x_i \leftarrow 1}, \sigma^i, j) = W(f_{x_i \leftarrow 0}, \sigma^i, j)$ for all $j \in [1\ldots n-1]$, then for all weighting factors $p \in (0,1)$, x_i does not have gain for (f, σ, p). Conversely,*

if $W(f_{x_i\leftarrow 1},\sigma^i,j)\neq W(f_{x_i\leftarrow 0},\sigma^i,j)$ *for some* $j\in[1\ldots n-1]$, *then for all but at most* $r-1$ *weighting factors* $p\in(0,1)$, x_i *has gain for* (f,σ,p).

Proof. Let f_0 denote $f_{x_i\leftarrow 0}$ and f_1 denote $f_{x_i\leftarrow 1}$. Let $\sigma\in\{0,1\}^n$ be an orientation.

For real valued variables y and z and for $a\in\{0,1\}^n$, let $T_{\sigma,a}(y,z)$ be the multiplicative term $y^{n-d}z^d$, where $d=\Delta(\sigma,a)$, the Hamming distance between σ and a. So, for example, if $\sigma=(1,1,1)$ and $a=(1,0,0)$, $T_{\sigma,a}(y,z)=yz^2$. Note that for $p\in(0,1)$, $T_{\sigma,a}(p,1-p)$ is the probability assigned to a by distribution $D_{(\sigma,p)}$. For $\sigma\in\{0,1\}^n$ and f a Boolean function on $\{0,1\}^n$, let $g_{f,\sigma}$ be the polynomial in y and z such that

$$g_{f,\sigma}(y,z)=\sum_{a\in\{0,1\}^n:f(a)=1}T_{\sigma,a}(y,z). \tag{5}$$

Thus, for example, if f is the two-variable disjunction $f(x_1,x_2)=x_1\vee x_2$, and $\sigma=(1,1)$, then $g_{f,\sigma}=y^1z^1+y^1z^1+y^2z^0=y^2+2yz$.

Define $g'(y,z)=g_{f_1,\sigma^i}(y,z)-g_{f_0,\sigma^i}(y,z)$, where g is as given in Equation 5. The quantity $W(f,\sigma,k)$ is the value of the coefficient of the term $y^{n-k}z^k$ in $g_{f,\sigma}$. Thus $g'(y,z)=\sum_{j=0}^{n-1}c_jy^{n-1-j}z^j$, where for all $j\in[0\ldots n-1]$, $c_j=W(f_1,\sigma^i,j)-W(f_0,\sigma^i,j)$.

Let $p\in(0,1)$. Under distribution $D_{(\sigma,p)}$, $\Pr(f=1|x_i=0)$ and $\Pr(f=1|x_i=1)$ are equal to $g_{f_0,\sigma^i}(p,1-p)$ and $g_{f_1,\sigma^i}(p,1-p)$ respectively. Thus by Lemma 1, x_i has gain for (f,σ,p) iff $g'(p,1-p)=0$.

Let $h(p)$ be the polynomial in p such that $h(p)=g'(p,1-p)$.

If x_i is irrelevant, then for all fixed $p\in(0,1)$, x_i has no gain for (f,σ,p). Further, $W(f_1,\sigma^i,j)=W(f_0,\sigma^i,j)$ for all $j\in[0\ldots n-1]$. Thus the lemma holds if x_i is irrelevant. In what follows, assume x_i is relevant.

If $W(f_1,\sigma^i,j)=W(f_0,\sigma^i,j)$ for all $j\in[0\ldots n-1]$, then $h(p)$ is identically 0 and for all fixed $p\in(0,1)$, x_i has no gain for (f,σ,p).

Suppose conversely that $W(f_1,\sigma^i,j)\neq W(f_0,\sigma^i,j)$ for some j. Then $g'(y,z)$ is not identically 0. We will show that $h(p)=g'(p,1-p)$ is a polynomial of degree at most $r-1$ that is not identically 0.

We begin by showing that $h(p)$ has degree at most $r-1$. Let $x_l\neq x_i$ be an irrelevant variable of f. Assume without loss of generality that $\sigma_l=1$. Then since $f(a_{x_l\leftarrow 1})=1$ iff $f(a_{x_l\leftarrow 0})=1$,

$$\begin{aligned}
g_{f,\sigma}(p,1-p) &= \sum_{a\in\{0,1\}^n:f(a)=1,a_l=1}pT_{\sigma^l,a^l}(p,1-p)+\sum_{a\in\{0,1\}^n:f(a)=1,a_l=0}(1-p)T_{\sigma^l,a^l}(p,1-p)\\
&= \sum_{a\in\{0,1\}^n:f(a)=1,a_l=0}T_{\sigma^l,a^l}(p,1-p)\\
&= \sum_{b\in\{0,1\}^{n-1}:f_{x_l\leftarrow 0}(b)=1}T_{\sigma^l,b}(p,1-p)\\
&= g_{f_{x_l\leftarrow 0},\sigma^l}(p,1-p).
\end{aligned}$$

That is, $g_{f,\sigma}(p,1-p)$ is equal to the corresponding polynomial for the function $g_{f_{x_l\leftarrow 0},\sigma^l}(p,1-p)$ produced by hardwiring irrelevant variable x_l to 0. By repeating this argument, we get that $g_{f,\sigma}=g_{\tilde{f},\tilde{\sigma}}$ where $\tilde{f}$ is the function obtained from f by hardwiring all of its irrelevant variables to 0, and $\tilde{\sigma}$ is σ restricted to the relevant variables of f. Thus g has degree at most r and $h(p)=g'(p,1-p)$ has degree at most $r-1$.

Let j' be the smallest j such that $W(f_1,\sigma^i,j) \neq W(f_0,\sigma^i,j)$. Then $c_{j'}$ is non-zero, and all (non-zero) terms of $g'(y,z)$ have the form $c_j y^{r-1-j} z^j$ where $j \geq j'$. We can thus factor out $z^{j'}$ from $g'(y,z)$ to get $g'(y,z) = z^{j'} g''(y,z)$, where $g''(y,z) = \sum_{j=j'}^{r-1} c_j y^{r-1-j} z^{j-j'}$. One term of g'' is $c_{j'} y^{r-1-j'}$, while all other terms have a non-zero power of z. Thus for $p=1$, $g''(p,1-p) = c_{j'}$ which is non-zero, proving that $g''(p,1-p)$ is not identically 0. Hence $h(p) = z^{j'} g''(p,1-p)$ is the product of two polynomials that are not identically 0, and so $h(p)$ is not identically 0.

Finally, since $h(p)$ is a polynomial of degree at most $r-1$ that is not identically 0, it has at most $r-1$ roots. It follows that there are at most $r-1$ values of p in $(0,1)$ such that x_i does not have gain for (f,σ,p). □

We now present the main theorem of this section.

Theorem 9.1 *Let f be a non-constant Boolean function on $\{0,1\}^n$. Let $\sigma \in \{0,1\}^n$ be an orientation, and let p be chosen uniformly at random from $(0,1)$. Then with probability 1 there exists at least one variable x_i such that x_i has gain for (f,σ,p).*

Proof. Let $\sigma \in \{0,1\}^n$ be a fixed orientation. Let r be the number of relevant variables of f. Let x_i be the variable of f whose existence is guaranteed by Lemma 8. Thus $W(f_{x_i \leftarrow 1},\sigma^i,j) \neq W(f_{x_i \leftarrow 0},\sigma^i,j)$ for some j. By Lemma 9, for all but at most $r-1$ weighting factors $p \in (0,1)$, x_i has gain for (f,σ,p). With probability 1, a random p chosen uniformly from $(0,1)$ will not be equal to one of those $r-1$ weighting factors. □

Using the techniques above, one can also show that for certain p-biased distributions $D[p]$, there do not exist any non-constant correlation immune functions with respect to $D[p]$. Let f be a non-constant Boolean function defined on $\{0,1\}^n$. By Lemma 8 and the proof of Lemma 9, there is some variable x_i such that associated polynomial $h(p)$ (defined with respect to $\sigma = (1,\ldots,1)$) is not identically 0. It follows that for any p that is not a root of h, x_i has gain for $(f,(1,\ldots,1),p)$, and thus f is not correlation immune with respect to distribution $D[p]$. The polynomial $h(p)$ has degree at most $n-1$ and integer coefficients with magnitude at most 2^n, which restricts its possible roots. For example, every root of h must be algebraic. Thus for any non-algebraic p, there are no Boolean functions that are correlation immune with respect to $D[p]$. Similarly, since h has integral coefficients with magnitude bounded by 2^n, an elementary theorem on polynomials (sometimes called the "Rational Zeroes Theorem") immediately implies that any rational zero of h must have magnitude at least $1/2^n$. Thus for any p such that $0 < p < 1/2^n$, there are no n-variable Boolean functions that are correlation immune with respect to $D[p]$.

With Theorem 9.1 we have shown that for any non-constant function and any orientation σ, there exists at least one variable x_i such that if p is chosen randomly, then, with probability 1, x_i has gain with respect to f under the distribution $D_{(\sigma,p)}$. However, the theorem says nothing about the magnitude of the gain. If the chosen p is close to a root of the polynomial $h(p)$, defined in the proof of Lemma 9, then the gain will be very small. Moreover, the gain can vary depending on the function and on the skew. (We will prove a result later in the paper, in Lemma 11, which shows that with a certain probability, a randomly chosen p will cause x_i to have reasonably large gain.)

The identity of the variable(s) having gain can also depend on the skew. There may be relevant variables other than x_i that don't have gain for any p. In the example given following the proof of Lemma 7, variables x_1 and x_2 will have no gain for $(f,(0,\ldots,0),p)$ no matter the choice of p.

Theorem 9.1 suggests that skewing is an effective method for finding relevant variables of a non-constant Boolean f, because for nearly all skews, there will be at least one variable with non-zero

gain. Equivalently, for nearly all skewed distributions, function f is not correlation immune with respect to that distribution. However, in practice—even in a noiseless situation where examples are all labeled correctly according to a function f—we do not usually have access to the entire truth table, and thus are not able to compute the exact gain of a variable under distribution $D_{(\sigma,p)}$ defined by the skew. We can only estimate that gain. Moreover, in practice we cannot sample from the distribution $D_{(\sigma,p)}$. Instead, we simulate $D_{(\sigma,p)}$ by reweighting our sample.

9.3 Analysis of Sequential Skewing

Sequential skewing is a variant of skewing. In sequential skewing, n iterations of reweighting are performed, where n is the number of input variables of the target function. On the j^{th} iteration, examples are reweighted according to the preferred setting of the j^{th} variable alone; if the setting of the j^{th} variable matches the preferred setting, the example is multiplied by p, otherwise the example is multiplied by $1-p$. The reweighting in the jth iteration is designed to simulate the product distribution in which each variable other than x_j is 1 with probability 1/2, and variable x_j has its preferred setting with probability p. In addition to the n iterations of reweighting, the gain of every variable is also calculated with respect to the original, unweighted, data set. As in standard skewing, the algorithm uses the calculated gains to determine which variable to output.

In the reweighting done by sequential skewing, there is a chosen variable x_i, a preferred setting $c \in \{0,1\}$ of that variable, and a weight factor $p \in (0,1)$. We thus define a (sequential) skew to be a triple (i,c,p), where $i \in [1 \ldots n]$, $c \in \{0,1\}$, and $p \in (0,1)$. Define the probability distribution $D_{(i,c,p)}$ on $\{0,1\}^n$ such that for $a \in \{0,1\}^n$, $D_{(i,c,p)}$ assigns probability $p \cdot (\frac{1}{2})^{n-1}$ to a if $a_i = c$, and $(1-p) \cdot (\frac{1}{2})^{n-1}$ otherwise. Thus $D_{(i,c,p)}$ is the distribution that would be generated by applying sequential skewing, with parameters x_i, c and p, to the entire truth table.

Let f be a Boolean function on $\{0,1\}^n$. We say that variable x_j *has gain for* (f,i,c,p) if under distribution $D_{(i,c,p)}$, $G(f|x_j) > 0$. By Lemma 1, x_j has gain for (i,c,p) iff under distribution $D_{(i,c,p)}$, $\Pr[f=1|x_j=1] \neq \Pr[f=1|x_j=0]$.

We will use the following lemma.

Lemma 10 *A Boolean function f is 2-correlation immune iff it is 1-correlation immune, and for all pairs $i<j$, the inputs x_i and x_j are independent given $f(x_1,\ldots,x_n)$.*

Proof. We first prove the forward direction. If f is 2-correlation immune, then it is certainly 1-correlation immune, and all triples (f,x_i,x_j) are mutually independent.

The reverse direction is a calculation. Let $\alpha,\beta,\gamma \in \{0,1\}$. Using pairwise independence, and then 1-correlation immunity, we get

$$\begin{aligned}\Pr[f=\alpha, x_i=\beta, x_j=\gamma] &= \Pr[f=\alpha]\Pr[x_i=\beta, x_j=\gamma \mid f=\alpha] \\ &= \Pr[f=\alpha]\Pr[x_i=\beta \mid f=\alpha]\Pr[x_j=\gamma \mid f=\alpha] \\ &= \Pr[f=\alpha]\Pr[x_i=\beta]\Pr[x_j=\gamma].\end{aligned}$$

This holds even if $\Pr[f=\alpha]=0$, for then both sides vanish. □

The constant functions $f=0$ and $f=1$ are 2-correlation immune, as is any parity function on 3 or more variables. We have enumerated the 2-correlation immune functions up to $n=5$ and found that when $n \leq 4$, the only such functions are as above, but for $n=5$, others begin to appear. Specifically, there are 1058 2-correlation immune functions of 5 variables, but only 128 parity

functions and complements of these (with no constraint on the relevant variables). (Our enumeration method works as follows. Vanishing of the relevant Fourier coefficients can be expressed as a linear system with 0-1 solutions, which we can count by a "splitting" process reminiscent of the time-space tradeoff for solving subset sum problems, Odlyzko 1980.) Denisov (1992) gave an asymptotic formula for the number of 2-correlation immune functions, and from this work it follows that for large n, only a small fraction of the 2-correlation immune functions will be parity functions.

The following theorem shows that, in our idealized setting, sequential skewing can identify a relevant variable of a function, unless that function is 2-correlation immune. It follows that sequential skewing will be ineffective in finding relevant variables of a parity function, even with unlimited sample sizes. In contrast, standard skewing can identify relevant variables of a parity function if the sample size is large enough.

Theorem 9.2 *Let f be a correlation-immune Boolean function on $\{0,1\}^n$, let $i \in [1\ldots n]$, and let $c \in \{0,1\}$. Let p be chosen uniformly at random from $(0,1)$. If the function f is 2-correlation immune, then for all $j \in [1\ldots n]$, x_j has no gain for (f,i,c,p). Conversely, if f is not 2-correlation immune, then for some $j \in [1\ldots n]$, x_j has gain for (f,i,c,p) with probability 1.*

Proof. Let f be a correlation immune function. Let $i \in [1\ldots n]$ and $c \in \{0,1\}$.

Assume $c = 1$. The proof for $c = 0$ is symmetric and we omit it. Consider skew (i,c,p), where $p \in (0,1)$. Let $f^{-1}(1) = \{x \in \{0,1\}^* | f(x) = 1\}$.

Let $j \in [1\ldots n]$. Let $A_1 = |\{a \in f^{-1}(1) \mid a_i = c \text{ and } a_j = 1\}|$, and $B_1 = |\{a \in f^{-1}(1) \mid a_i \neq c \text{ and } a_j = 1\}|$. Similarly, let $A_0 = |\{a \in f^{-1}(1) \mid a_i = c \text{ and } a_j = 0\}|$, $B_0 = |\{a \in f^{-1}(1) \mid a_i \neq c \text{ and } a_j = 0\}|$.

Under distribution $D_{(i,c,p)}$, if $j \neq i$, $\Pr[f = 1 | x_j = 1] = (A_1 p + B_1(1-p)) \left(\frac{1}{2}\right)^{n-2}$. If $j = i$, then because $c = 1$, $\Pr[f = 1 | x_j = 1] = A_1 \left(\frac{1}{2}\right)^{n-1}$. Similarly, if $j \neq i$, $\Pr[f = 1 | x_j = 0] = (A_0 p + B_0(1-p)) \left(\frac{1}{2}\right)^{n-2}$. If $j = i$, $\Pr[f = 1 | x_j = 0] = B_0 \left(\frac{1}{2}\right)^{n-1}$.

The difference $\Pr[f = 1 | x_j = 1] - \Pr[f = 1 | x_j = 0]$ is a linear function in p. If $i \neq j$, this function is identically zero iff $A_1 = A_0$ and $B_1 = B_0$. If it is not identically 0, then there is at most one value of $p \in (0,1)$ for which it is 0. If $i = j$, this function is identically zero iff $A_1 = B_0$. Also note that for $i = j$, $A_0 = 0$ and $B_1 = 0$ by definition.

In addition, since f is correlation immune, $A_1 + A_0 = B_1 + B_0$. If $i = j$, then $\Pr[f = 1 | x_j = 1] - \Pr[f = 1 | x_j = 0]$ is therefore identically zero and x_i has no gain for (f,i,c,p). If $j \neq i$, then x_j has no gain for (f,i,c,p) iff $A_1 = A_0 = B_1 = B_0$. This latter condition is precisely the condition that $\Pr[x_i = \alpha \wedge x_j = \beta | f = \gamma] = \Pr[x_i = \alpha | f = \gamma] \Pr[x_j = \beta | f = \gamma]$ under the uniform distribution on $\{0,1\}^n$. If this condition holds for all pairs $i \neq j$, no variable x_j has gain for (f,i,c,p), and by Lemma 10, f is 2-correlation immune. Otherwise for some $i \neq j$, x_j has gain for (f,i,c,p) for all but at most 1 value of p. The theorem follows. □

10. Exploiting Product Distributions

Until now we have *simulated* alternative product distributions through skewing. But simulating alternative distributions is not the same as sampling directly from them. In particular, skewing can magnify idiosyncracies in the sample in a way that does not occur when sampling from true alternative distributions. We now consider the PDC model, in which the learning algorithm can specify product distributions and request random examples from those distributions. In practice it might be

possible to obtain examples from such alternative distributions by working with a different population or varying an experimental set-up. Intuitively, one might expect a high degree of overhead in making such changes, in which case it would be desirable to keep the number of alternative distributions small.

10.1 FindRel1: Finding a Relevant Variable Using r Distributions

Let Boolean$_{r,n}$ denote the Boolean functions on n variables that have at most r relevant variables. We first present a simple algorithm that we call FindRel1, based on Theorem 9.1. It identifies a relevant variable of any target function in Boolean$_{r,n}$, with probability $1-\delta$, by estimating the first-order Fourier coefficient of x_i for r distinct product distributions. The algorithm assumes that r is known. If not, standard techniques can be used to compensate. For example, one can repeat the algorithm with increasing values of r (perhaps using doubling), until a variable is identified as being relevant.

The algorithm works as follows. For $j \in \{1,\ldots,r\}$, let D_j denote the product distribution that sets each of the n input variables to 1 with probability $j/(r+1)$. For each D_j, the algorithm requests a sample S_j of size m_0 (we will specify m_0 in the proof below). Then, for each of the n input variables x_i, it estimates the associated first-order Fourier coefficients from sample S_j by computing $\hat{f}_{S,D_j}(x_i)$. At the end, the algorithm outputs the set of all variables x_i whose gain on some S_j exceeded a threshold θ_0 (also specified below).

Theorem 10.1 *For all non-constant $f \in$ Boolean$_{r,n}$, with probability at least $1-\delta$ FindRel1 will output a non-empty subset of the relevant variables of f. FindRel1 uses a total of $O((r+1)^{2r}\ln\frac{2nr}{\delta})$ examples, drawn from r distinct p-biased distributions. The running time of FindRel1 is polynomial in $2^{r\ln r}$, n, and $\ln\frac{1}{\delta}$.*

Proof. Since f is non-constant, it has at least one relevant variable. Recall that for distribution D on $\{0,1\}^n$, $G_D(f,x_i)$ denotes the gain of x_i with respect to f under distribution D. Recall also that $D[p]$ denotes the product distribution that sets each variable x_i to 1 with probability p.

By the arguments in Section 9, for each relevant variable x_i, $\Pr_{D[p]}[f=1|x_i=1]-\Pr_{D[p]}[f=1|x_i=0]$ can be written as a polynomial of degree $r-1$ in p. Call this polynomial $h_i(p)$. For all irrelevant variables x_i of f, $h_i(p)$ is identically 0.

Now let x_i be a relevant variable such that $h_i(p)$ is not identically 0. By Theorem 9.1, f has at least one such relevant variable. The polynomial $h_i(p)$ has degree at most $r-1$ and hence has at most $r-1$ roots. Therefore, for at least one $j \in \{1,\ldots,r\}$, $h_i(j/(r+1)) \neq 0$.

Let $j^* \in \{1,\ldots,r\}$ be such that $h_i(j^*/(r+1)) \neq 0$. Since h_i has integer coefficients and is of degree at most $r-1$, it follows that $h_i(j^*/(r+1)) = b/(r+1)^{r-1}$, for some integer b. Thus the absolute value of $h_i(j^*/(r+1))$ is at least $1/(r+1)^{r-1}$, and by Lemma 2, the first-order Fourier coefficient (for distribution D_{j^*}) associated with x_i has magnitude at least $2\frac{\sqrt{\frac{j^*}{(r+1)}(1-\frac{j^*}{r+1})}}{(r+1)^{(r-1)}}$, which is lower bounded by $q := 2\frac{\sqrt{\frac{1}{(r+1)}(1-\frac{1}{r+1})}}{(r+1)^{(r-1)}}$. Set θ_0 in the description of FindRel1 to be $q/2 = \sqrt{r}/(r+1)^r$.

For any single D_j, it follows from Lemma 3 that if $m_0 = 2(r+1)^{2r}r^{-1}\ln\frac{2nr}{\delta}$, if we use a sample of size m_0 drawn from D_j and estimate all n first-order Fourier coefficients for distribution D_j using that sample, then with probability at least $1-\frac{\delta}{r}$, each of the estimates will have additive error less than $q/2$. Thus with probability at least $1-\delta$, this will hold for all r of the D_j. The total number of examples drawn by FindRel1 is $rm_0 = 2(r+1)^{2r}\ln\frac{2nr}{\delta}$.

Since for some relevant variable, the associated Fourier coefficient is at least q for some D_j, and for all irrelevant variables, the associated Fourier coefficient is 0 for all D_j, the theorem follows. □

Skewing uses gain estimates, rather than estimates of the first-order Fourier coefficients. FindRel1 can be modified to use gain estimates. By a similar argument as above, it follows from Lemma 1 that for distribution D_{j^*}, some relevant variable has gain at least $q' = 2\frac{1}{r+1}(1-\frac{1}{r+1})(\frac{1}{r+1})^{2r-2}$ with respect to that distribution. We could thus modify FindRel1 to output the variables whose gain exceeds $q'/2$. Then Lemma 6 implies that a sample of size $m_0 = O(r^{4r-2}\ln\frac{nr}{\delta})$ would suffice for the modified FindRel1 to output a non-empty subset of relevant variables. This sample complexity bound is higher than the bound for the original FindRel1 based on Fourier coefficients.

10.2 FindRel2: Lowering the Sample Complexity

We now present our second algorithm, FindRel2. As discussed in the introduction, it has an advantage over FindRel1 in terms of running time and sample complexity, but requires examples from a larger number of distinct distributions. FindRel2 is based on the following lemma.

Lemma 11 *Let f have $r \geq 1$ relevant variables. Suppose p is chosen uniformly at random from $(0,1)$. Then there exists a relevant variable x_i of f, and a value $\tau \geq 2e^{-3r}$ such that with probability at least $\tau/2$ (with respect to the choice of p), $G_{D[p]}(f,x_i) \geq \tau/2$.*

Proof By Theorem 9.1 and its proof, there exists a variable x_i of f such that $\Pr_{D[p]}[f=1|x_i=1] - \Pr_{D[p]}[f=1|x_i=0]$ can be expressed as a polynomial $h_i(p)$, which has integer coefficients and is not identically 0. Let $g(p) = G_{D[p]}(f,x_i)$. By Lemma 1,

$$g(p) = 2p(1-p)h_i(p)^2.$$

Then there are integers $\gamma_0,\ldots,\gamma_{2r}$ such that $g(p) = 2\sum_{j=0}^{2r}\gamma_j p^j$. Since $g(p)$ is non-negative but not identically 0, we have

$$\tau := \int_0^1 g(p)dp = 2\sum_{j=0}^{2r}\frac{\gamma_j}{j+1} > 0.$$

This is at least $2/L$, where L is the least common multiple of $\{1,\ldots,2r+1\}$. Observe that for each prime, the number of times it appears in the prime factorization of L equals the number of its powers that are $\leq 2r+1$. By an explicit form of the prime number theorem,

$$\log L = \sum_{\substack{p^k \leq 2r+1 \\ k\geq 1}} \log p \leq 3r.$$

(This can be checked directly for $r=1$, and for $r \geq 2$ we can use Theorem 12 of Rosser and Schoenfeld 1962.) Thus, $\tau \geq 2e^{-3r}$. Now let α be the fraction of $p \in (0,1)$ for which $g(p) \geq \tau/2$. Then,

$$\tau = \int_{g\geq\tau/2} g + \int_{g<\tau/2} g \leq \alpha + (\tau/2)(1-\alpha).$$

This implies $\alpha \geq \tau/(2-\tau) > \tau/2$, and the lemma follows. □

Note that the proof of the above lemma relies crucially on the non-negativity of the gain function, and thus the same proof technique could not be applied to first-order Fourier coefficients, which can be negative.

It is possible that the bounds in the above result could be improved by exploiting how g comes from the Boolean function f. Without such information, however, the bounds are essentially the best possible. Indeed, by properly choosing g, one can use this idea to estimate the density of primes from below, and get within a constant factor of the prime number theorem. See Montgomery (1994) for a discussion of this point.

FindRel2, our second algorithm for finding a relevant variable, follows easily from the above lemma. We describe the algorithm in terms of two size parameters m_1 and m_2, and a classification threshold θ_1.

The algorithm begins by choosing m_1 values for p, uniformly at random from $(0,1)$. Let P be the set of chosen values. For each value $p \in P$, the algorithm requests m_2 random examples drawn with respect to distribution $D[p]$, forming a sample S_p. Then, for each of the n input variables x_i, it computes $G(S_p,x_i)$, the gain of x_i on the sample S_p. At the end, the algorithm outputs all variables x_i such that $G(S_p,x_i) > \theta_1$ for at least one of the generated samples S_p.

Using Lemma 11, we can give values to parameters m_1, m_2, and θ_1 in FindRel2 and prove the following theorem.

Theorem 10.2 *For all non-constant $f \in Boolean_{r,n}$, with probability at least $1-\delta$, FindRel2 will output a non-empty subset of the relevant variables of f. FindRel2 uses $O(e^{9r}(r+\ln(n/\delta))\ln(1/\delta))$ examples, drawn from $O(e^{3r}\log\frac{1}{\delta})$ product distributions. The running time is polynomial in 2^r, n, and $\log\frac{1}{\delta}$.*

Proof. As in the proof of Theorem 10.1, f has at least one relevant variable x_i for which $h_i(p)$ is not identically 0. Let x_{i^*} denote this variable. Let $\delta_1 = \delta_2 = \delta/2$.

If the statement of Lemma 11 holds for any value of τ at all, it holds for the lower bound. We therefore let $\tau = 2e^{-3r}$. By Lemma 11, for at least a $\tau/2$ fraction of the values of $p \in (0,1)$, $G_{D[p]}(f,x_{i^*}) \geq \tau/2$. Let us call these "good" values of p. If a single p is chosen uniformly at random from $(0,1)$, then the probability p is good is at least $\tau/2$.

Let $m_1 = e^{3r}\ln\frac{1}{\delta_1} = \frac{2}{\tau}\ln\frac{1}{\delta_1}$. If the algorithm chooses m_1 independent random values of p to form the set P, the probability that P does not contain any good p's is at most $(1-\tau/2)^{m_1} \leq e^{-m_1\tau/2} = \delta_1$.

Suppose P contains at least one good p. Let p^* be such a p. Let $\gamma = G_{D[p^*]}(f,x_{i^*})$. Then, $\gamma \geq \tau/2 = e^{-3r}$. Set θ_1 in the algorithm to $e^{-3r}/2$, the resulting lower bound for $\gamma/2$.

Set m_2 in the algorithm to be equal to $256\ln(2nm_1/\delta_2)/\theta_1^2$.

Consider any $p \in P$. Then by Lemma 6, with probability at least $1-\delta_2/m_1$, $|G(S_p,x_i) - G_{D[p]}(x_i)| < \gamma/2$ for all variables x_i. Since $|P| = m_1$, it follows that $|G(S_p,x_i) - G_{D[p]}(x_i)| < \gamma/2$ holds for all variables x_i and for all $p \in P$, with probability at least $1-\delta_2$.

Assuming P has at least one good p^*, $G_{D[p^*]}(x_{i^*}) \geq \gamma$, while for all $p \in P$ and all irrelevant x_i, and $G_{D[p]}(x_i) = 0$. Thus if $|G(S_p,x_i) - G_{D[p]}(x_i)| < \gamma/2$ holds for every x_i and $p \in P$, and P contains at least one good p, then FindRel2 outputs a non-empty subset of relevant variables of f.

It follows that the the probability that the algorithm does not output a non-empty subset of the relevant variables is at most $\delta_1 + \delta_2 = \delta$, as claimed.

It remains to estimate the number of examples used, which is $m_1 m_2$. The only problem is with m_2. Since $0 < \delta_1 < 1/2$, we have $0 < \ln(2\ln(1/\delta_1)) < \ln(1/\delta_1)$. Using this, together with the definitions of m_1 and τ, we find that

$$\ln(2nm_1/\delta_2) \quad = \quad \ln(2n) + \ln(2\ln(1/\delta_1)) - \ln(\tau) - \ln(\delta_2)$$

$$\begin{aligned} &\leq \ln(n) + \ln(1/\delta_1) + 3r + \ln(1/\delta_2) \\ &= \ln(n) + 3r + 2\ln(2/\delta). \end{aligned}$$

Combining this with the definitions of m_2 and θ_1 gives us $m_2 = O(e^{6r}(r + \ln(n/\delta)))$, and since $m_1 = e^{3r}\ln(2/\delta)$, we get $m_1 m_2 = O(e^{9r}(r + \ln(n/\delta))\ln(1/\delta))$. □

We do not know the best exponents for which a result like Theorem 10.2 is true. We do note, however, that more careful use of the prime number theorem would allow the exponents 9 and 3 to be lowered to $6 + o(1)$ and $2 + o(1)$, respectively.

Using not too many more examples, the random choices can be eliminated from FindRel2, as follows. Since the g appearing in the proof of Lemma 11 is a polynomial, the set of $p \in [0,1]$ for which $g(p) \geq \tau/2$ is a finite union of closed intervals. Their lengths sum to at least $\tau/2 = e^{-3r}$. In the open interval between any two adjacent closed intervals, there must be a local minimum of g, which is a zero of g', a polynomial of degree $\leq 2r - 1$. It follows that there are at most $2r$ of these closed intervals, making one have length at least $h := e^{-3r}/(2r)$. Our algorithm can therefore try $p = h, 2h, 3h, \ldots$ and be guaranteed that one of these is good. (We don't have to try $p = 0, 1$ because g vanishes there.) With this modification, the number of distributions becomes $O(re^{3r})$ and the number of examples becomes $O(re^{9r}(r + \ln(n/\delta)))$.

10.3 Two Algorithms From The Literature

Another approach to finding a relevant variable is implicit in work of Bshouty and Feldman (2002). We present it briefly here.

Bshouty and Feldman's approach is based on the following facts. Variable x_i is relevant to f iff there is some Fourier coefficient $\hat{f}(z)$ with $z_i = 1$ and $\hat{f}(z) \neq 0$. Further, if f has r relevant variables, the absolute value of every non-zero Fourier coefficient of f is at least $1/2^r$.

For $b \in \{0,1\}^{n-1}$, let $1b$ denote the concatenation of 1 with b. Let $w(b)$ denote the Hamming weight of b. Define $R_1(f) = \sum_{b \in \{0,1\}^{n-1}} \hat{f}^2(1b)(\frac{1}{2^{2w(b)}})$. Thus R_1 is a weighted sum of the Fourier coefficients $\hat{f}(z)$ such that $z_1 = 1$. For any $z \in \{0,1\}^n$, the quantity $\hat{f}^2(z)$ is non-zero only if $\{i | z_i = 1\} \subseteq \{i |$ variable x_i is a relevant variable of $f\}$. Therefore, if $\hat{f}^2(1b) \neq 0$, then $w(b) \leq r$. It follows that if x_1 is relevant, $R_1 > 1/2^{4r}$. If x_1 is irrelevant, $R_1 = 0$. Let D' be the product distribution specified by the parameter vector $[1/2, 1/4, 1/4, \ldots, 1/4]$ and let $w \in \{0,1\}^n$ be such that $w = [1, 0, \ldots, 0]$. As shown by Bshouty and Feldman (2002, proof of Lemma 11), $R_1 = E_{x \sim U}[E_{y \sim D'}[f(y)\chi_w(x \oplus y)]]^2$. Here $x \sim U$ denotes that the first expectation is with respect to an x drawn from the uniform distribution on $\{0,1\}^n$, and $y \sim D'$ denotes that the second expectation is with respect to a y drawn from distribution D'. For any fixed x, $E_{y \sim D'}[f(y)\chi_w(x \oplus y)]]$ can be estimated by drawing random samples $(y, f(y))$ from D'. The quantity R_1 can thus be estimated by uniformly generating values for x, estimating $E_{y \sim D'}[f(y)\chi_w(x \oplus y)]]$ for each x, and then taking the average over all generated values of x. Using arguments of Bshouty and Feldman, which are based on a standard Hoeffding bound, it can be shown that for some constant c_1, a sample of size $O(2^{c_1 r}\log^2(\frac{1}{\delta'}))$ from D' suffices to estimate R_1 to within an additive error of $\frac{1}{2^{4r+1}}$, with probability $1 - \delta'$. If the estimate obtained is within this error, then whether x_i is relevant can be determined by just checking whether the estimate is greater than $\frac{1}{2^{4r+1}}$. We can apply this procedure to all n variables x_i, each time taking a sample of y's from a new distribution. Setting $\delta' = \delta/n$, it follows that a sample of size $O(n2^{c_1 r}\log^2\frac{n}{\delta})$ suffices to determine, with probability $1 - \delta$, which of the n variables are relevant. Thus this algorithm finds all the relevant variables.

The above algorithm uses examples chosen from n product distributions. Each product distribution has exactly one parameter set to $1/2$, and all other parameters set to a fixed value $\rho \neq 1/2$ (here $\rho = 1/4$, although this choice was arbitrary).

If the parameters of the product distribution can be set to 0 and 1, membership queries can be simulated. We now briefly describe an algorithm that uses membership queries and uniform random examples to find a relevant variable of a target function with at most r relevant variables. A similar approach is used in a number of algorithms for related problems (see, e.g., Arpe and Reischuk, 2007; Guijarro et al., 1999; Blum et al., 1995; Damaschke, 2000; Bshouty and Hellerstein, 1998).

The algorithm first finds the value of $f(a)$ for some arbitrary a, either by asking a membership query or choosing a random example. Then, the algorithm draws a random sample S (with respect to the uniform distribution) of size $2^r \ln \frac{1}{\delta}$. Assuming the function contains at least one relevant variable, a random example has probability at least $1/2^r$ of being negative, and probability at least $1/2^r$ of being positive. Thus if the function has at least 1 relevant variable, with probability at least $1-\delta$, S contains an example a' such that $f(a') \neq f(a)$. (If it contains no such example, the algorithm outputs the constant function $f(x) = f(a)$.) The algorithm then takes a and a', and using membership queries, executes a standard binary-search procedure for finding a relevant variable of a Boolean function, given a positive and a negative example of that function (cf. Blum et al., 1995, Lemma 4). This procedure makes $O(\log n)$ membership queries.

If we carry out the membership queries in the PDC model by asking for examples from product distributions with parameters 0 and 1, the result is an algorithm that finds a relevant variable with probability at least $1-\delta$ using $O(\log n)$ product distributions and $O(2^r \log \frac{1}{\delta})$ random examples. The random examples can also be replaced by membership queries on (n,r) universal sets (see, e.g., Bshouty and Hellerstein, 1998).

11. On the Limitations of Skewing

One of the motivating problems for skewing was that of learning the parity of r of n variables. The results of Section 9 imply that skewing is effective for learning parity functions if the entire truth table is available as the training set. (Of course, if the entire truth table is available, there are much more straightforward ways of identifying relevant variables.) Equivalently, we can identify relevant variables if we are able to determine the exact gain of each variable with respect to skewed distributions. In practice, though, we need to estimate gain values based on a random sample. The random sample must be large enough so that we can still identify a relevant variable, even though the gain estimates for the variables will have some error. We now consider the following sample complexity question: how large a random sample is needed so that skewing can be used to identify a relevant variable of the parity function, with "high" probability? We would like to know how quickly this sample complexity grows as r and n grow.

Skewing is not a statistical query learning algorithm, but it is based on the estimation of statistics. In what follows, we use techniques that were previously employed to prove lower bounds for statistical query learning of parity functions.

It is difficult to analyze the behavior of skewing because the same sample is used and re-used for many gain calculations. This introduces dependencies between the resulting gain estimates. Here we consider a modification of the standard skewing procedure, in which we pick a new, independent random sample each time we estimate the gain of a variable with respect to a skew (σ, p). We call this modification "skewing with independent samples." Intuitively, since the motivation behind

skewing is based on estimating statistical quantities, choosing a new sample to make each estimate should not hurt accuracy. In experiments, skewing with independent samples was more effective in finding relevant variables than standard skewing (Ray et al., 2009).

For simplicity, assume that the variable output by the skewing algorithm is one that exceeds a fixed threshold the maximum number of times. However, as we discuss below, our lower bounds would also apply to implementations using other output criteria.

We prove a sample complexity lower bound for skewing with independent samples, when applied to a target function that is the parity of r of n variables. The proof is based on the fact that the skewing algorithm does not use all the information in the examples. Given a skew (σ, p), and an example $(x, f(x))$, the skewing algorithm weights this example according to $d = \Delta(x, \sigma)$, the Hamming distance between x and σ. The calculation of the gain for a variable x_i on the weighted data set then depends only on $f(x)$, whether $x_i = \sigma_i$, and on d. These three pieces of information together constitute a "summary" of the example $(x, f(x))$, for orientation σ. The skewing algorithm uses only these summaries; it does not use any other information about the examples. We will argue that the summaries do not contain enough information to identify relevant variables of a parity function, unless the sample size is "large".

We begin by proving a technical lemma, using techniques of Jackson (2003) and Blum et al. (1994).

Let $\text{Parity}_{r,n}$ be the set of parity functions on n variables which have r relevant variables. So for each $f \in \text{Parity}_{r,n}$, $f(x_1, \ldots, x_n) = x_{i_1} + x_{i_2} + \ldots + x_{i_r}$ where the sum is taken mod 2, and the x_{i_j} are distinct. Let $NEQ(b,c)$ denote the inequality predicate, that is, $NEQ(b,c) = 1$ if $b \neq c$ and $NEQ(b,c) = 0$ if $b = c$.

Let $d \in \{0, \ldots, n\}$ and $b, c \in \{0,1\}$. For $f \in \text{Parity}_{r,n}$ and $\sigma \in \{0,1\}^n$, the quantity $\Pr[NEQ(\sigma_i, x_i) = b, f(x) = c, \text{ and } \Delta(x,\sigma) = d]$ has the same value for all relevant variables x_i of f (where the probability is with respect to the uniform distribution over all $x \in \{0,1\}^n$). The same holds for all irrelevant variables x_i of f. We define $S_1^{f,\sigma}(b,c,d) = \Pr[NEQ(\sigma_i, x_i) = b, f(x) = c, \text{ and } \Delta(x,\sigma) = d]$ when x_i is a relevant variable of f, and $S_2^{f,\sigma}(b,c,d) = \Pr[NEQ(\sigma_i, x_i) = b, f(x) = c, \text{ and } \Delta(x,\sigma) = d]$ when x_i is an irrelevant variable of f.

As an example, suppose $\sigma' \in \{0,1\}^n$ is such that $f(\sigma') = 0$. Then $S_1^{f,\sigma'}(0,1,d) = \frac{1}{2^n} \sum_{t \in T} \binom{r-1}{t} \binom{n-r}{d-t}$ where $T = \{t \in \mathbb{Z} | t \text{ is odd and } 0 \leq t \leq d\}$. Similarly, $S_2^{f,\sigma'}(1,0,d) = \frac{1}{2^n} \sum_{t \in T'} \binom{r}{t} \binom{n-r-1}{d-1-t}$ where $T' = \{t \in \mathbb{Z} | t \text{ is even and } 0 \leq t \leq d-1\}$.

For variable x_i and orientation σ, we call $(NEQ(\sigma_i, x_i), f(x), \Delta(x,\sigma))$ the *summary tuple* corresponding to $(x, f(x))$. Thus for target function $f \in \text{Parity}_{r,n}$ and orientation σ, $S_1^{f,\sigma}(b,c,d)$ is the probability of obtaining a summary tuple (b,c,d) for variable x_i when x_i is relevant, and $S_2^{f,\sigma}(b,c,d)$ is the same probability in the case that x_i is irrelevant.

We prove the following upper bound on $|S_1^{f,\sigma}(b,c,d) - S_2^{f,\sigma}(b,c,d)|$.

Lemma 12 *For all $\sigma \in \{0,1\}^n$, $f \in \text{Parity}_{r,n}$, $b, c \in \{0,1\}$ and $d \in \{0, \ldots, n\}$,*

$$|S_1^{f,\sigma}(b,c,d) - S_2^{f,\sigma}(b,c,d)| \leq \frac{1}{2} \left(\binom{n-1}{r}^{-1/2} + \binom{n-1}{r-1}^{-1/2} \right)$$

.

Proof. Suppose first that $f(\sigma) = 0$. For any $\sigma' \in \{0,1\}^n$ such that $f(\sigma') = 0$, $S_1^{f,\sigma'}(b,c,d) = S_1^{f,\sigma}(b,c,d)$, and the analogous equality holds for S_2. Without loss of generality, we may therefore assume that $\sigma = 0^n$.

Let $S_1 = S_1^{f,\sigma}(b,c,d)$ and $S_2 = S_2^{f,\sigma}(b,c,d)$. Let $\gamma = |S_1 - S_2|$. Define a function $\psi_i(x,y) : \{0,1\}^n \times \{0,1\} \to \{1,-1\}$ such that $\psi_i(x,y) = -1$ if $NEQ(\sigma_i, x_i) = b$, $y = c$, and $\Delta(x,\sigma) = d$, and $\psi_i(x,y) = 1$ otherwise.

For x_i a relevant variable of f, $E[\psi_i(x,f(x))] = 1 - 2S_1$ (where the expectation is with respect to the uniform distribution on $x \in \{0,1\}^n$). Similarly, for x_i an irrelevant variable of f, $E[\psi_i(x,f(x))] = 1 - 2S_2$.

Let x_j be a relevant variable of f, and let x_k be an irrelevant variable of f.

Since $|S_1 - S_2| = \gamma$,

$$|E[\psi_j(x,f(x))] - E[\psi_k(x,f(x))]| = 2|S_1 - S_2| = 2\gamma.$$

As noted by Jackson (2003), it follows from an analysis in Blum et al. (1994) that for any parity function h on n variables, and any function $g : \{0,1\}^{n+1} \to \{1,-1\}$,

$$E[g(x,h(x))] = \hat{g}(0^{n+1}) + \hat{g}(z1)$$

where $z \in \{0,1\}^n$ is the characteristic vector of the relevant variables of h (equivalently, $\chi_z = 1 - 2h$), and $z1$ denotes the assignment $(z_1,\ldots,z_n,1)$.

Thus we have

$$E[\psi_j(x,f(x))] = \hat{\psi}_j(0^{n+1}) + \hat{\psi}_j(z1)$$
$$E[\psi_k(x,f(x))] = \hat{\psi}_k(0^{n+1}) + \hat{\psi}_k(z1)$$

where z is the characteristic vector of the relevant variables of f. It follows from the definition of ψ_i that $\hat{\psi}_j(0^{n+1}) = \hat{\psi}_k(0^{n+1})$. Therefore,

$$|\hat{\psi}_j(z1) - \hat{\psi}_k(z1)| = 2\gamma.$$

Now consider any other parity function $f' \in \text{Parity}_{r,n}$. Since $\sigma = 0^n$, $f'(\sigma) = f(\sigma) = 0$. Therefore, $S_1^{f',\sigma} = S_1$ and $S_2^{f',\sigma} = S_2$. If relevant variable x_j of f is also a relevant variable of f', then $E[\psi_j(x,f'(x))] = \hat{\psi}_j(0^{n+1}) + \hat{\psi}_j(z'1)$, where z' is the characteristic vector of the relevant variables of f'. Thus $\hat{\psi}_j(z'1) = \hat{\psi}_j(z1)$.

There are $\binom{n-1}{r-1}$ functions $f' \in \text{Parity}_{r,n}$ such that x_j is a relevant variable of f'. It follows that there are at least $\binom{n-1}{r-1}$ Fourier coefficients of ψ_j that are equal to $\hat{\psi}_j(z1)$. By Parseval's identity,

$$|\hat{\psi}_j(z1)| \leq \binom{n-1}{r-1}^{-1/2}.$$

Similarly, $E[\psi_k(x,f(x))] = E[\psi_k(x,f'(x))]$ for all $f' \in \text{Parity}_{r,n}$ such that x_k is an irrelevant variable of f'. Since there are $\binom{n-1}{r}$ such f', an analogous argument shows that

$$|\hat{\psi}_k(z1)| \leq \binom{n-1}{r}^{-1/2}.$$

Thus

$$\begin{aligned}\gamma &= \frac{|\hat{\psi}_j(z1) - \hat{\psi}_k(z1)|}{2} \\ &\leq \frac{|\hat{\psi}_j(z1)| + |\hat{\psi}_k(z1)|}{2} \\ &\leq \frac{1}{2}\left(\binom{n-1}{r}^{-1/2} + \binom{n-1}{r-1}^{-1/2}\right).\end{aligned}$$

Thus the lemma holds in the case that $f(\sigma) = 0$.

Now suppose that $f(\sigma) = 1$. Given $a \in \{0,1\}^n$, $f(a) = 1$ iff a differs from σ in an even number of its relevant variables (and in an arbitrary number of its irrelevant variables). Further, $f(a) = 1$ iff a differs from 0^n in an odd number of its relevant variables (and in an arbitrary number of its irrelevant variables). Thus $S_1^{f,\sigma}(b,c,d) = S_1^{f,0^n}(b,1-c,d)$ and $S_2^{f,\sigma}(b,c,d) = S_2^{f,0^n}(b,1-c,d)$.

Since the bound proved above for the case $f(\sigma) = 0$ holds for arbitrary c, it holds for $|S_1^{f,0^n}(b,1-c,d) - S_2^{f,0^n}(b,1-c,d)|$, and the lemma follows. □

The above lemma gives an upper bound on $\gamma = |S_1^{f,\sigma}(b,c,d) - S_2^{f,\sigma}(b,c,d)|$. Another way to prove such an upper bound is to use the fact that a statistical query algorithm could determine whether variable x_i was relevant by asking a query requesting the value of $\Pr[NEQ(\sigma_i,x_i) = b, f(x) = c, \text{ and } \Delta(x,\sigma) = d]$ within tolerance $\gamma/2$ (assuming $\gamma > 0$). Queries of this type could be used to find all the relevant variables of f, which uniquely determines parity function f. If γ were too large, this would contradict known lower bounds on statistical learning of parity. This approach yields a bound that is close to the one given in the lemma above, but the proof is less direct. (See, for example, Blum et al. 1994 for the definition of the statistical query model.)

We now prove a sample complexity lower bound for learning parity functions, using skewing with independent samples.

Theorem 11.1 *Suppose we use skewing with independent samples to identify a relevant variable of f, where $f \in Parity_{r,n}$. Assuming that the samples are drawn from the uniform distribution, to successfully output a relevant variable with probability at least μ requires that the total number of examples used in making the gain estimates be at least $\frac{(\mu - \frac{r}{n})\min\{\binom{n-1}{r-1}^{1/2}, \binom{n-1}{r}^{1/2}\}}{4(n+1)}$.*

Proof. Consider running skewing with independent samples with a target function $f \in \text{Parity}_{r,n}$. To estimate the gain of a variable x_i with respect to a skew (σ, p), the skewing algorithm uses a sample drawn from the uniform distribution. In calculating this estimate, the algorithm does not use the full information in the examples. For each labeled example $(x, f(x))$, it uses only the information in the corresponding summary tuple $(b,c,d) = (NEQ(\sigma_i,x_i), f(x), \Delta(x,\sigma))$. We may therefore assume that the skewing algorithm is, in fact, given only the summary tuples, rather than the raw examples.

The number of distinct possible summary tuples is at most $4(n+1)$, since there are two possible values each for b and c, and $n+1$ possible values for d. The uniform distribution on examples x induces a distribution D on the summary tuples generated for skew (σ, p) and variable x_i. For fixed σ, distribution D is the same for all relevant variables x_i of f. It is also the same for all irrelevant variables x_i of f. Let D_1^σ be the distribution for the relevant variables, and D_2^σ be the distribution for

the irrelevant variables. Let q be the distance between D_1^σ and D_2^σ as measured in the L_1 norm. That is, if K denotes the set of possible summary tuples, then $q = \sum_{z \in K} |\Pr_{D_1^\sigma}[z] - \Pr_{D_2^\sigma}[z]|$.

Since there are at most $4(n+1)$ possible summary tuples. it follows from Lemma 12 that $q \leq 2(n+1)(\binom{n-1}{r-1}^{-1/2} + \binom{n-1}{r}^{-1/2})$.

Let m be the total number of examples used to estimate the gain of all variables x_i under all skews (σ, p) used by the skewing algorithm. Since the L_1 distance between D_1^σ and D_2^σ is at most q for every skew (σ, p) and every variable x_i, it follows that during execution of the algorithm, with probability at least $(1-q)^m$, the summary tuples generated for the relevant variables of f are distributed in the same way as the summary tuples generated for the irrelevant variables of f.

By the symmetry of the parity function, if the target function f is randomly chosen from $\text{Parity}_{r,n}$, then with probability at least $(1-q)^m$, the final variable output by the skewing algorithm when run on this f is equally likely to be any of the n input variables of f. Thus the probability that the skewing algorithm outputs an irrelevant variable is at least $(1-q)^m(\frac{n-r}{n})$, and the probability that it outputs a relevant variable is at most $1-(1-q)^m(\frac{n-r}{n}) < 1-(1-qm)(1-\frac{r}{n}) < \frac{r}{n}+qm(1-\frac{r}{n}) < \frac{r}{n}+qm$. The first inequality in this sequence holds because $(1-q)^m \geq (1-qm)$, since $0 < q < 1$.

Since the above holds for a random target function in $\text{Parity}_{r,n}$, it holds for the worst-case $f \in \text{Parity}_{r,n}$. It follows that if skewing with independent samples outputs a relevant variable of f (for any $f \in \text{Parity}_{r,n}$) with probability at least μ, then the total number of examples used must be at least $\frac{\mu - \frac{r}{n}}{q}$. Since $q \leq 2(n+1)(\binom{n-1}{r-1}^{-1/2} + \binom{n-1}{r}^{-1/2})$, it follows that $1/q \geq \frac{\min\{\binom{n-1}{r-1}^{1/2}, \binom{n-1}{r}^{1/2}\}}{4(n+1)}$. □

To make the theorem concrete, consider the case where $r = \log n$. Note that if we simply choose one of the n variables at random, the probability of choosing a relevant variable in this case is $\frac{\log n}{n}$. It follows from the theorem that for skewing to output a relevant variable with success "noticeably" greater than random guessing, that is, with probability at least $\frac{logn}{n} + \frac{1}{p(n)}$, for some polynomial p, it would need to use more than a superpolynomial number of examples.

The above proof relies crucially on the fact that skewing uses only the information in the summary tuples. The details of how the summary tuples are used is not important to the proof. Thus the lower bound applies not only to the implementation of skewing that we assumed (in which the chosen variable is the one whose gain exceeds the fixed threshold the maximum number of times). Assuming independent samples, the lower bound would also apply to other skewing implementations, including, for example, an implementation in which the variable with highest gain over all skews was chosen as the output variable.

On the other hand, one can also imagine variants of skewing to which the proof would not apply. For example, suppose that we replaced the single parameter p used in skewing by a vector of parameters $[p_1, \ldots, p_n]$, so that in reweighting an example, variable x_i causes the weight to be multiplied by either p_i or $1-p_i$, depending on whether there is a match with x_i's preferred setting. Our proof technique would not apply here, since we would be using information not present in the summary tuples. To put it another way, the proof exploits the fact that the distributions used by skewing are simple ones, defined by a pair (σ, p). Interestingly, it was our focus on such simple distributions that led us to the two new algorithms in Section 10.

The negative result above depends on the fact that for f a parity function with r relevant variables, the distribution of the summary tuples for a relevant variable x_i is very close to the distribution of the summary tuples for an irrelevant variable x_i. For other correlation immune functions, the distributions are further apart, making those functions easier for skewing to handle. For example, consider $\text{Consensus}_{r,n}$, the set of all n-variable Boolean functions with r relevant variables, whose

value is 1 iff the r relevant variables are all equal. The functions in this set are correlation immune. Assume $n+r$ is even. Let $d=(n+r)/2$ and $\sigma=(1,1,\ldots,1)$. Let $S_1=\Pr[x_i=0,\Delta(x,\sigma)=d,$ and $f(x)=1]$ when x_i is a relevant variable of f. Let $S_2=\Pr[x_i=0,\Delta(x,\sigma)=d,$ and $f(x)=1]$ when x_i is an irrelevant variable of f. Then $S_1=\frac{1}{2^n}\binom{n-r}{\frac{n-r}{2}}$ and $S_2=\frac{1}{2^n}(\binom{n-r-1}{\frac{n-r}{2}-1}+\binom{n-r-1}{\frac{n+r}{2}-1})$. Then $S_1-S_2=\Omega(\frac{1}{2^n}\binom{n-r}{\frac{n-r}{2}})$, since the first term of S_2 is equal to $S_1/2$, and the second term of S_2 is much smaller than the first. Since $\binom{m}{m/2}=\theta(\frac{2^m}{\sqrt{m}})$, $S_1-S_2=\Omega(\frac{1}{\sqrt{n-r}2^r})$. Even for r as large as $n/2$, this is $\Omega(\frac{1}{\sqrt{n}2^r})$. Note the difference between this quantity and the analogous bound for parity. The dependence here is on $\frac{1}{2^r}$ rather than on roughly $\binom{n}{r}^{1/2}$.

12. Conclusions and Open Questions

In this paper, we studied methods of finding relevant variables that are based on exploiting product distributions.

We provided a theoretical study of skewing, an approach to learning correlation immune functions (through finding relevant variables) that has been shown empirically to be quite successful. On the positive side, we showed that when the skewing algorithm has access to the complete truth table of a target Boolean function—a case in which standard greedy gain-based learners fail—skewing will succeed in finding a relevant variable of that function. More particularly, under any random choice of skewing parameters, a single round of the skewing procedure will find a relevant variable with probability 1.

In some sense the correlation immune functions are the hardest Boolean functions to learn, and parity functions are among the hardest of these to learn, since a parity function of $k+1$ variables is k-correlation immune. In contrast to the positive result above, we showed (using methods from statistical query learning) that skewing needs a sample size that is superpolynomial in n to learn parity of $\log n$ relevant variables, given examples from the uniform distribution.

We leave as an open question the characterization of the functions of $\log n$ variables that skewing can learn using a sample of size polynomial in n, given examples from the uniform distribution.

Skewing operates on a sample from a single distribution, and can only *simulate* alternative product distributions. We used the PDC model to study how efficiently one can find relevant variables, given the ability to sample directly from alternative product distributions. We presented two new algorithms in the PDC model for identifying a relevant variable of an n-variable Boolean function with r relevant variables.

We leave as an open problem the development of PDC algorithms with improved bounds, and a fuller investigation of the tradeoffs between time and sample complexity, and the number and types of distributions used. As a first step, it would be interesting to show an algorithm whose time complexity is polynomial in n when $r=\log n$, using a number of p-biased distributions that is polynomial in $\log n$. Our lower bound for parity relied on the assumption of independent samples. We suspect that the lower bound also holds if the assumption is removed, but proving it seems to require a different approach. As we mentioned earlier, it is a major open problem whether there is a polynomial-time algorithm for finding relevant variables of a function of $\log n$ variables, using only examples from the uniform distribution.

Acknowledgments

David Page, Soumya Ray, and Lisa Hellerstein gratefully aknowledge support from the National Science Foundation (NSF IIS 0534908). Eric Bach gratefully acknowledges support from the National Science Foundation (NSF CCF-0523680 and CCF-0635355) and from a Vilas Research Associate Award from the Wisconsin Alumni Research Foundation. We thank Matt Anderson for letting us use his counts of 2-correlation immune functions and Jeff Jackson for answering questions about the statistical query literature. Part of this work was performed while Lisa Hellerstein was visiting the University of Wisconsin, Madison.

References

T. Akutsu, S. Miyano, and S. Kuhara. A simple greedy algorithm for finding functional relations: Efficient implementation and average case analysis. *Theor. Comput. Sci.*, 292(2):481–495, 2003. doi: http://dx.doi.org/10.1016/S0304-3975(02)00183-4.

N. Alon. Derandomization via small sample spaces (abstract). In *SWAT '96: Proceedings of the 5th Scandinavian Workshop on Algorithm Theory*, pages 1–3, 1996.

J. Arpe and E. Mossel. Application of a generalization of Russo's formula to learning from multiple random oracles. *Combinatorics, Probability and Computing*, to appear. Published online by Cambridge University Press 09 Jul 2009, doi:10.1017/S0963548309990277. Preliminary version published at http://arxiv.org/abs/0804.3817, 2008.

J. Arpe and R. Reischuk. When does greedy learning of relevant attributes succeed? In *COCOON '07: Proceedings of the 13th Annual International Conference on Computing and Combinatorics*, volume 4598 of *Lecture Notes in Computer Science*, pages 296–306. Springer, 2007.

E. Bach. Improved asymptotic formulas for counting correlation immune Boolean functions. *SIAM J. Discrete Math.*, to appear. Preliminary version appeared as Technical Report Number 1616, Computer Sciences Dept., University of Wisconsin–Madison, 2007.

A. Blum. Learning a function of r relevant variables. In *COLT/Kernel '03: Learning Theory and Kernel Machines, Proceedings of the 16th Annual Conference on Learning Theory and 7th Kernel Workshop*, Lecture Notes In Artificial Intelligence: Vol. 2777, pages 731–733. Springer Verlag, 2003.

A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *STOC '94: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, pages 253–262, 1994.

A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. *J. Comput. Syst. Sci.*, 50(1):32–40, 1995.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

L. Brynielsson. A short proof of the Xiao-Massey lemma. *IEEE Transactions on Information Theory*, 35(6):1344–1344, 1989.

N. H. Bshouty and V. Feldman. On using extended statistical queries to avoid membership queries. *J. Mach. Learn. Res.*, 2:359–395, 2002.

N. H. Bshouty and L. Hellerstein. Attribute-efficient learning in query and mistake-bound models. *J. Comput. Syst. Sci.*, 56(3):310–319, 1998.

P. Camion, C. Carlet, P. Charpin, and N. Sendrier. On correlation-immune functions. In *CRYPTO '91: Advances in Cryptology*, pages 86–100. Springer-Verlag, 1991.

P. Damaschke. Adaptive versus nonadaptive attribute-efficient learning. *Machine Learning*, 41(2): 197–215, 2000.

O. V. Denisov. An asymptotic formula for the number of correlation-immune of order k Boolean functions. *Discrete Math. Appls.*, 2(4):407–426, 1992.

V. Feldman, P. Gopalan, S. Khot, and A. K. Ponnuswami. New results for learning noisy parities and halfspaces. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 563–574, 2006.

D. Fukagawa and T. Akutsu. Performance analysis of a greedy algorithm for inferring Boolean functions. *Inf. Process. Lett.*, 93(1):7–12, 2005. doi: http://dx.doi.org/10.1016/j.ipl.2004.09.017.

M. L. Furst, J. C. Jackson, and S. W. Smith. Improved learning of AC^0 functions. In *COLT '91: Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 317–325, 1991.

S. W. Golomb. On the classification of boolean functions. *IRE Transactions on Information Theory*, IT-5:176–186, 1959.

S. W. Golomb. On the cryptanalysis of nonlinear sequences. In *IMA – Cryptography and Coding '99*, Lecture Notes In Computer Science: Vol. 1746, pages 236–242. Springer-Verlag, 1999.

D. Guijarro, J. Tarui, and T. Tsukiji. Finding relevant variables in PAC model with membership queries. In *ALT '99: Proceedings of the 10th International Conference on Algorithmic Learning Theory*, 1999.

J. Jackson. On the efficiency of noise-tolerant PAC algorithms derived from statistical queries. *Annals of Math. and Artificial Intell.*, 39(3):291–313, 2003.

E. Lantz, S. Ray, and D. Page. Learning Bayesian network structure from correlation immune data. In *UAI '07: Proceedings of the 23rd International Conference on Uncertainty in Artificial Intelligence*, 2007.

Y. Mansour. Learning Boolean functions via the Fourier transform. *Theoretical Advances in Neural Computation and Learning*, pages 391–424, 1994.

H. L. Montgomery. *Ten Lectures on the Interface Between Analytic Number Theory and Harmonic Analysis*. AMS, 1994.

E. Mossel, R. O'Donnell, and R. A. Servedio. Learning juntas. In *STOC '03: Proceedings of the 35th Annual Symposium on the Theory of Computing*, pages 206–212, 2003.

A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In *Cryptology and Computational Number Theory: Proceedings of Symposia in Applied Mathematics*, volume 42, pages 79–88. AMS, 1980.

D. Page and S. Ray. Skewing: An efficient alternative to lookahead for decision tree induction. In *IJCAI: Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 601–612, 2003.

E. M. Palmer, R. C. Read, and R. W. Robinson. Balancing the n-cube: a census of colorings. *J. Algebraic Combin.*, 1:257–273, 1992.

I. Pinelis. An approach to inequalities for the distributions of infinite-dimensional martingales. In R. M. Dudley, M. G. Hahn, and J. Kuelbs, editors, *Probability in Banach Spaces*, pages 128–134. Birkhauser, 1992.

J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1997.

S. Ray and D. Page. Sequential skewing: An improved skewing algorithm. In *ICML '04: Proceedings of the 21st International Conference on Machine Learning*, 2004.

S. Ray and D. Page. Generalized skewing for functions with continuous and nominal variables. In *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*, pages 705–712, 2005.

S. Ray, E. Lantz, B. Rosell, L. Hellerstein, and D. Page. Learning correlation immune functions by skewing: An empirical evaluation. Unpublished manuscript, 2009.

B. Rosell, L. Hellerstein, S. Ray, and D. Page. Why skewing works: Learning difficult functions with greedy tree learners. In *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*, pages 728–735, 2005.

J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6:64–94, 1962.

B. Roy. A brief outline of research on correlation immune functions. In *Proceedings of the 7th Australian Conference on Information Security and Privacy*, Lecture Notes In Computer Science: Vol. 2384, pages 379–384. Springer Verlag, 2002.

T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, IT-30(5):776–780, 1984.

L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

G.-Z. Xiao and J. L. Massey. A spectral characterization of correlation-immune combining functions. *IEEE Transactions on Information Theory*, 34(3):569–570, 1988.

K. Yang. On learning correlated boolean functions using statistical queries. In *ALT '01: Proceedings of the 12th International Conference on Algorithmic Learning Theory*, volume 2225 of *Lecture Notes in Computer Science*, pages 59–76. Springer, 2001.

K. Yang. New lower bounds for statistical query learning. *J. Comput. Syst. Sci.*, 70(4):485–509, 2005.

Journal of Machine Learning Research 10 (2009) 2413-2444 Submitted 11/06; Revised 12/08; Published 11/09

Reinforcement Learning in Finite MDPs: PAC Analysis

Alexander L. Strehl[*] ASTREHL@FACEBOOK.COM
Facebook
1601 S California Ave.
Palo Alto, CA 94304

Lihong Li[†] LIHONG@YAHOO-INC.COM
Yahoo! Research
4401 Great America Parkway
Santa Clara, CA 95054

Michael L. Littman MLITTMAN@CS.RUTGERS.EDU
Department of Computer Science
Rutgers University
Piscataway, NJ 08854

Editor: Sridhar Mahadevan

Abstract

We study the problem of learning near-optimal behavior in finite Markov Decision Processes (MDPs) with a polynomial number of samples. These "PAC-MDP" algorithms include the well-known E^3 and R-MAX algorithms as well as the more recent Delayed Q-learning algorithm. We summarize the current state-of-the-art by presenting bounds for the problem in a unified theoretical framework. A more refined analysis for upper and lower bounds is presented to yield insight into the differences between the model-free Delayed Q-learning and the model-based R-MAX.

Keywords: reinforcement learning, Markov decision processes, PAC-MDP, exploration, sample complexity

1. Introduction

In the reinforcement-learning (RL) problem (Sutton and Barto, 1998), an agent acts in an unknown or incompletely known environment with the goal of maximizing an external reward signal. In the most standard mathematical formulation of the problem, the environment is modeled as a finite Markov Decision Process (MDP) where the goal of the agent is to obtain near-optimal discounted return. Recent research has dealt with probabilistic bounds on the number of samples required for near-optimal learning in finite MDPs (Kearns and Singh, 2002; Kakade, 2003; Brafman and Tennenholtz, 2002; Strehl and Littman, 2005; Strehl et al., 2006a,b). The purpose of this paper is to summarize this field of knowledge by presenting the best-known upper and lower bounds for the problem. For the upper bounds, we present constructive proofs using a unified framework in Section 3.1; these tools may be useful for future analysis. While none of the bounds we present are entirely new, the main contribution of this paper is to streamline as well as consolidate their

∗. Some of this work was completed while A. Strehl was at Rutgers University and also while he was at Yahoo! Research.

†. Some of this work was completed while L. Li was at Rutgers University.

analyses. In addition, the bounds we present are stated in terms of an *admissible* heuristic provided to the algorithm (see Section 1.3) and the (unknown) optimal value function. These bounds are more refined than the ones previously presented in the literature and more accurately reflect the performance of the corresponding algorithms. For the lower bound, we provide an improved result that matches the upper bound in terms of the number of states of the MDP.

An outline of the paper is as follows. This introduction section concludes with a formal specification of the problem and related work. In Section 2, R-MAX and Delayed Q-learning are described. Then, we present their analyses and prove PAC-MDP upper bounds in Section 3. A new lower bound is proved in Section 4.

1.1 Main Results

We present two upper bounds and one lower bound on the achievable *sample complexity* of general reinforcement-learning algorithms (see Section 1.5 for a formal definition). The two upper bounds dominate all previously published bounds, but differ from one another. When logarithmic factors are ignored, the first bound, for the R-MAX algorithm, is

$$\tilde{O}(S^2A/(\varepsilon^3(1-\gamma)^6)),$$

while the corresponding second bound, for the Delayed Q-learning algorithm, is

$$\tilde{O}(SA/(\varepsilon^4(1-\gamma)^8)).$$

Here, S and A are the number of states and the number of actions, respectively, of the MDP, ε and δ are accuracy parameters, and γ is a discount factor. R-MAX works by building an approximate MDP model and the S^2A term in its sample complexity follows from requiring accuracy in each of the S^2A parameters of the model. Delayed Q-learning, on the other hand, does not build an explicit model and can be viewed as an approximate version of value iteration. Thus, accuracy only needs to be guaranteed for each of the SA entries in the value function.

While previous bounds are in terms of an upper bound $1/(1-\gamma)$ on the value function, we find that tighter bounds are possible if a more informative value-function upper bound is given. Specifically, we can rewrite the bounds in terms of the initial admissible heuristic values (see Section 1.3) supplied to the algorithms, $U(\cdot,\cdot)$, and the true (unknown) value function $V^*(\cdot)$. Ignoring logarithmic factors, for R-MAX the bound is

$$\tilde{O}\left(\frac{V_{\max}^3 S\,|\{(s,a)\in\mathcal{S}\times\mathsf{A}|U(s,a)\geq V^*(s)-\varepsilon\}|}{\varepsilon^3(1-\gamma)^3}\right), \tag{1}$$

and for Delayed Q-learning

$$\tilde{O}\left(\frac{V_{\max}^3 \sum_{(s,a)\in\mathcal{S}\times\mathsf{A}}[U(s,a)-V^*(s)]_+}{\varepsilon^4(1-\gamma)^4}\right), \tag{2}$$

where $V_{\max}\geq \max_{s,a}U(s,a)$ is an upper bound on the admissible heuristic (and also on the true value function), and $[x]_+$ is defined as $\max(0,x)$ for $x\in\mathbb{R}$. Thus, we observe that for R-MAX one factor of $SA/(1-\gamma)^3$ gets replaced by $|\{(s,a):U(s,a)\geq V^*(s)-\varepsilon\}|V_{\max}^3$,[1] the number of state-action pairs whose heuristic initial value is larger than $V^*-\varepsilon$, while for Delayed Q-learning the

1. This quantity can be as small as $SV_{\max}^3$ and as large as $SAV_{\max}^3$, where $V_{\max}\in[0,\frac{1}{1-\gamma}]$.

factor $SA/(1-\gamma)^4$ is replaced by $V_{\max}^3 \sum_{(s,a)\in S\times A}(U(s,a)-V^*(s))$,[2] $V_{\max}^3$ times the total sum of differences between the heuristic values and the optimal value function. The latter term is better, because it takes more advantage of accurate heuristics. For instance, if $U(s,a)=V^*(s)+\varepsilon$ and $V^*(s)$ is large for all s, then the bound for R-MAX stays essentially the same but the one for Delayed Q-learning is greatly improved. Please see Russell and Norvig (1994) for discussions and references on admissible heuristics. The method of incorporating admissible heuristics into Q-learning (Ng et al., 1999) and R-MAX (Asmuth et al., 2008) are well known, but the bounds given in Equation 1 and Equation 2 are new.

The upper bounds summarized above may be pessimistic and thus may not reflect the worst-case behavior of these algorithms. Developing lower bounds, especially *matching* lower bounds, tells us what can (or cannot) be achieved. Although matching lower bounds are known for deterministic MDPs (Koenig and Simmons, 1996; Kakade, 2003), it remains an open question for general MDPs. The previous best lower bound is due to Kakade (2003), and was developed for the slightly different notion of H-horizon value functions instead of the γ-discounted ones we focus on here. Adapting his analysis to discounted value functions, we get the following lower bound:

$$\Omega\left(\frac{SA}{\varepsilon(1-\gamma)^2}\ln\frac{1}{\delta}\right).$$

Based on the work of Mannor and Tsitsiklis (2004), we provide an improved lower bound

$$\Omega\left(\frac{SA}{\varepsilon^2}\ln\frac{S}{\delta}\right) \tag{3}$$

which simultaneously increases the dependence on both S and $1/\varepsilon$. While we choose to drop dependence on $1/(1-\gamma)$ in our lower bound to facilitate a cleaner analysis, we believe it is possible to force a quadratic dependence by a more careful analysis. This new lower bound (3) has a few important implications. First, it implies that Delayed Q-learning's worst-case sample complexity has the *optimal* dependence on S. Second, it increases the dependence on $1/\varepsilon$ significantly from linear to quadratic. It would be interesting to know whether a cubic dependence on $1/\varepsilon$ is possible, which would match the upper bound for R-MAX (ignoring logarithmic factors).

Our lower bound is tight for the factors S, $1/\varepsilon$, and $1/\delta$, in the weaker *parallel sampling* model (Kearns and Singh, 1999). This finding suggests that a worse dependence on $1/\varepsilon$ is possible only in MDPs with slow *mixing* rates.[3] In both the parallel sampling model and the MDP used to prove the lower bound given by Equation 3 (see Section 4), the distribution of states being sampled/visited mixes extremely fast (in one and two timesteps, respectively). The slower the mixing rate, the more difficult the *temporal credit assignment* problem (Sutton and Barto, 1998). In other words, a worse dependence on $1/\varepsilon$ may require the construction of an MDP where *deep planning* is necessary.

Before finishing the informal introduction, we should point out that the present paper focuses on *worst-case* upper bounds and so the sample complexity of exploration bounds like Equations 1 and 2 can be too conservative for MDPs encountered in practice. However, the algorithms and their analyses have proved useful for guiding development of more practical exploration schemes as well as improved algorithms. First of all, these algorithms formalize the principle of "optimism under the

2. This quantity can be as small as 0 and as large as $SAV_{\max}^4$, where $V_{\max}\in[0,\frac{1}{1-\gamma}]$.

3. There are many ways to define a mixing rate. Roughly speaking, it measures how fast the distribution of states an agent reaches becomes independent of the initial state and the policy being followed.

face of uncertainty" (Brafman and Tennenholtz, 2002) which has been empirically observed to be effective for encouraging active exploration (Sutton and Barto, 1998). Sample complexity analysis not only shows soundness of this principle in a mathematically precise manner, but also motivates novel RL algorithms with efficient exploration (e.g., Nouri and Littman 2009 and Li et al. 2009). Second, there are several places in the proofs where the analysis can be tightened under various assumptions about the MDP. The use of admissible heuristic functions as discussed above is one example; another example is the case where the number of next states reachable from any state-action pair is bounded by a constant, implying the factor S in Equation 1 may be shaved off (cf., Lemma 14). More opportunities lie in MDPs with various structural assumptions. Examples include factored-state MDPs (Kearns and Koller, 1999; Strehl et al., 2007; Diuk et al., 2009), Relocatable Action Models (Leffler et al., 2007), and Object-Oriented MDPs (Walsh et al., 2009), in all of which an exponential reduction in sample complexity can be achieved, as well as in MDPs where prior information about the model is available (Asmuth et al., 2009). Third, the streamlined analysis we present here is very general and applies not only to finite MDPs. Similar proof techniques have found useful in analyzing model-based algorithms for continuous-state MDPs whose dynamics are linear (Strehl and Littman, 2008a) or multivariate normal (Brunskill et al., 2008); see Li (2009) for a survey.

1.2 Markov Decision Processes

This section introduces the Markov Decision Process (MDP) notation used throughout the paper; see Sutton and Barto (1998) for an introduction. Let $\mathcal{P}_X$ denote the set of probability distributions over the set X. A finite MDP M is a five tuple $\langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is a finite set called the state space, A is a finite set called the action space, $T : \mathcal{S} \times \mathsf{A} \to \mathcal{P}_{\mathcal{S}}$ is the transition distribution, $\mathcal{R} : \mathcal{S} \times \mathsf{A} \to \mathcal{P}_{\mathbb{R}}$ is the reward distribution, and $0 \leq \gamma < 1$ is a discount factor on the summed sequence of rewards. We call the elements of $\mathcal{S}$ and A states and actions, respectively, and use S and A to denote the number of states and the number of actions, respectively. We let $T(s'|s,a)$ denote the transition probability of state s' of the distribution $T(s,a)$. In addition, $R(s,a)$ denotes the expectation of the distribution $\mathcal{R}(s,a)$.

We assume that the learner (also called the *agent*) receives S, A, and γ as input. The learning problem is defined as follows. The agent always occupies a single state s of the MDP M. The agent is told this state and must choose an action a. It then receives an *immediate reward* $r \sim \mathcal{R}(s,a)$ and is transported to a *next state* $s' \sim T(s,a)$. This procedure then repeats forever. The first state occupied by the agent may be chosen arbitrarily. Intuitively, the solution or goal of the problem is to obtain as large as possible reward in as short as possible time. In Section 1.5, we provide one possible formalization of this objective within the PAC-MDP framework. We define a *timestep* to be a single interaction with the environment, as described above. The t^{th} timestep encompasses the process of choosing the t^{th} action. We also define an *experience* of state-action pair (s,a) to refer to the event of taking action a from state s.

A *policy* is any strategy for choosing actions. A stationary policy is one that produces an action based on only the current state, ignoring the rest of the agent's history. We assume (unless noted otherwise) that rewards[4] all lie in the interval $[0,1]$. For any policy π, let $V_M^\pi(s) = \mathbf{E}[\sum_{j=1}^{\infty} \gamma^{j-1} r_j | s]$ ($Q_M^\pi(s,a) = \mathbf{E}[\sum_{j=1}^{\infty} \gamma^{j-1} r_j | s,a]$) denote the discounted, infinite-horizon value (action-value) func-

4. It is easy to generalize, by linear transformations (Ng et al., 1999), to the case where the rewards are bounded above and below by known but arbitrary constants without changing the optimal policy.

tion for π in M (which may be omitted from the notation) from state s. If H is a positive integer, let $V_M^\pi(s,H)$ denote the H-step value of policy π from s. If π is non-stationary, then s is replaced by a *partial path* $c_t = (s_1, a_1, r_1, \ldots, s_t)$, in the previous definitions. Specifically, let s_t and r_t be the t^{th} encountered state and received reward, respectively, resulting from execution of policy π in some MDP M. Then, $V_M^\pi(c_t) = \mathbf{E}[\sum_{j=0}^{\infty} \gamma^j r_{t+j} | c_t]$ and $V_M^\pi(c_t, H) = \mathbf{E}[\sum_{j=0}^{H-1} \gamma^j r_{t+j} | c_t]$. These expectations are taken over all possible infinite paths the agent might follow in the future. The optimal policy is denoted π^* and has value functions $V_M^*(s)$ and $Q_M^*(s,a)$. Note that a policy cannot have a value greater than $1/(1-\gamma)$ by the assumption that the maximum reward is 1.[5]

1.3 Admissible Heuristics

We also assume that the algorithms are given an admissible heuristic for the problem before learning occurs. An **admissible heuristic** is a function $U : \mathcal{S} \times \mathsf{A} \to \mathbb{R}$ that satisfies $U(s,a) \geq Q^*(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathsf{A}$. We also assume that $U(s,a) \leq V_{\max}$ for all $(s,a) \in \mathcal{S} \times \mathsf{A}$ and some quantity $V_{\max}$. Prior information about the problem at hand can be encoded into the admissible heuristic and its upper bound $V_{\max}$. With no prior information, we can always set $U(s,a) = V_{\max} = 1/(1-\gamma)$ since $V^*(s) = \max_{a \in \mathsf{A}} Q^*(s,a)$ is at most $1/(1-\gamma)$. Therefore, without loss of generality, we assume $0 \leq U(s,a) \leq V_{\max} \leq 1/(1-\gamma)$ for all $(s,a) \in \mathcal{S} \times \mathsf{A}$.

1.4 A Note on the Use of Subscripts

Each algorithm that we consider maintains several variables. For instance, an *action value* or *action-value estimate*, $Q(s,a)$, sometimes called a *Q-value*, where (s,a) is any state-action pair, is maintained. We will often discuss a particular instance or time t during the execution of the algorithm. In this case, when we refer to $Q(s,a)$ we mean the value of that variable at the current moment. To be more explicit, we may write $Q_t(s,a)$, which refers to the value of $Q(s,a)$ immediately preceding the t^{th} action of the agent. Thus, $Q_1(s,a)$ is the initial value of $Q(s,a)$.

1.5 PAC-MDP Model

There are three essential ways to quantify the performance of a reinforcement-learning algorithm. They are *computational complexity*, the amount of per-timestep computation the algorithm uses during learning; *space complexity*, the amount of memory used by the algorithm; and *learning complexity*, a measure of how much experience the algorithm needs to learn in a given task. The last of these is difficult to define and several different ideas have been discussed in the literature. On the one hand, requiring an algorithm to "optimally explore"—meaning to obtain maximum expected discounted reward ($\mathbf{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t]$) over a known prior of MDPs—is an extremely difficult task tractable only in highly specialized cases (Gittins, 1989). Thus, we consider the relaxed but still challenging and useful goal of acting near-optimally on all but a polynomial number of steps (Kakade, 2003; Strehl and Littman, 2008b).

To formalize the notion of "efficient learning", we allow the learning algorithm to receive two additional inputs, ε and δ, both positive real numbers. The first parameter, ε, controls the quality of behavior we require of the algorithm (how close to optimality do we want the algorithm to be) and the second parameter, δ, is a measure of confidence (how certain do we want to be of the algorithm's

5. Thus, when comparing our results to the original R-MAX paper Brafman and Tennenholtz (2002), note that 1 takes the place of the quantity *Rmax*.

performance). As these parameters approach zero, greater exploration and learning is necessary, as higher quality is demanded of the algorithms.

In the following definition, we view an algorithm as a non-stationary (in terms of the current state) policy that, on each timestep, takes as input an entire history or trajectory through the MDP (its actual history) and outputs an action (which the agent then executes). Formally, we define the policy of any algorithm $\mathcal{A}$ at a fixed instance in time t to be a function $\mathcal{A}_t : \{\mathcal{S} \times \mathsf{A} \times [0,1]\}^* \times \mathcal{S} \to \mathsf{A}$, that maps future paths to future actions.[6]

Definition 1 *(Kakade 2003) Let $c = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots)$ be a random path generated by executing an algorithm $\mathcal{A}$ in an MDP M. For any fixed $\varepsilon > 0$, the* **sample complexity of exploration** *(**sample complexity**, for short) of $\mathcal{A}$ is the number of timesteps t such that the policy at time t, $\mathcal{A}_t$, satisfies $V^{\mathcal{A}_t}(s_t) < V^*(s_t) - \varepsilon$.*

Note that the sample complexity of an algorithm is dependent on some infinite-length path through the MDP. We believe this definition captures the essence of measuring learning. It directly measures the number of times the agent acts poorly (quantified by ε) and we view "fast" learners as those that act poorly as few times as possible. Based on this intuition, we define what it means to be an "efficient" learning algorithm.

Definition 2 *An algorithm $\mathcal{A}$ is said to be an* **efficient PAC-MDP** *(Probably Approximately Correct in Markov Decision Processes) algorithm if, for any $\varepsilon > 0$ and $0 < \delta < 1$, the per-timestep computational complexity, space complexity, and the sample complexity of $\mathcal{A}$ are less than some polynomial in the relevant quantities $(S, A, 1/\varepsilon, 1/\delta, 1/(1-\gamma))$, with probability at least $1-\delta$. It is simply* **PAC-MDP** *if we relax the definition to have no computational complexity requirement.*

The terminology, PAC, in this definition is borrowed from Angluin (1988) for the distribution-free supervised-learning model of Valiant (1984). One thing to note is that we only require a PAC-MDP algorithm to behave poorly (non-ε-optimally) on no more than a small (polynomially) number of timesteps. We do not place any limitations on when the algorithm acts poorly or how poorly it acts on those timesteps. This definition is in contrast to Valiant's PAC notion, which is more "off-line" in that it requires the algorithm to make all of its mistakes ahead of time (during the learning phase) before identifying a near-optimal policy. The notion of PAC-MDP is also closely related to the Mistake Bound (MB) model of Littlestone (1988) where the goal of a learner that predicts sequentially must make a small (polynomial) number of mistakes during a whole run. Indeed, if we count every timestep in which an algorithm behaves non-ε-optimally as a mistake, then a PAC-MDP algorithm makes only a polynomial number of mistakes during a whole run with high probability, similar to an MB algorithm. However, a mistake in a PAC-MDP algorithm refers to the quality of a policy rather than prediction errors as in MB.

Efficient learnability in the sample-complexity framework from above implies efficient learnability in a more realistic framework called *Average Loss* that measures the actual return (sum of rewards) achieved by the agent against the expected return of the optimal policy (Strehl and Littman, 2008b). The analysis of R-MAX by Kakade (2003) and of MBIE by Strehl and Littman (2005) use the same definition as above. The analysis of R-MAX by Brafman and Tennenholtz (2002) and of

6. The action of an agent on timestep t in state s_t is given by the function evaluated at the empty history, $\mathcal{A}_t(\emptyset, s_t)$.

E^3 by Kearns and Singh (2002) use slightly different definitions of efficient learning.[7] Our analyses are essentially equivalent, but simpler in the sense that mixing-time arguments are avoided. Compared with recently published regret bounds (Auer et al., 2009), our sample complexity bounds are easier to obtain and do not depend on quantities like mixing time or diameter that may be hard to determine *a priori*.

1.6 Related Work

There has been some theoretical work analyzing RL algorithms. In a Bayesian setting, with a known prior over possible MDPs, we could ask for the policy that maximizes expected reward. This problem has been solved (Gittins, 1989) for a specialized class of MDPs, called K-armed bandits. However, a solution to the more general problem seems unlikely to be tractable, although progress has been made (Duff and Barto, 1997; Poupart et al., 2006).

Early results include proving that under certain conditions various algorithms can, in the limit, compute the optimal value function from which the optimal policy can be extracted (Watkins and Dayan, 1992). These convergence results make no performance guarantee after only a finite amount of experience, although more recent work has looked at convergence rates (Szepesvári, 1998; Kearns and Singh, 1999; Even-Dar and Mansour, 2003). These types of analyses make assumptions that simplify the exploration issue.

The work by Fiechter (1994) was the first to prove that efficient (polynomial) approximate learning is achievable, via a model-based algorithm, when the agent has an action that *resets* it to a distinguished start state. Other recent work has shown that various model-based algorithms, including E^3 (Kearns and Singh, 2002), R-MAX (Brafman and Tennenholtz, 2002), and MBIE (Strehl and Littman, 2005) can achieve polynomial learning guarantees without the necessity of resets.

2. Algorithms

The total number of RL algorithms introduced in the literature is huge, so we limit the study to those with the best formal PAC-MDP learning-time guarantees. The two algorithms we study are R-MAX and Delayed Q-learning, because the best sample complexity bounds known for any PAC-MDP algorithm are dominated by the bound for one of these two algorithms. However, the bounds for R-MAX and Delayed Q-learning are incomparable—the bound for R-MAX is better in terms of $1/\varepsilon$ and $1/(1-\gamma)$, while the bound for Delayed Q-learning is better in terms of S. In fact, in Section 4 we will show that the sample complexity of Delayed Q-learning is optimal in terms of S via a matching lower bound.

2.1 R-MAX

Suppose that the agent has acted for some number of timesteps and consider its experience with respect to some fixed state-action pair (s,a). Let $n(s,a)$ denote the number of timesteps in which the agent has taken action a from state s. Suppose the agent has observed the following $n(s,a)$ immediate rewards for taking action a from state s: $r[1],r[2],\ldots,r[n(s,a)]$. Then, the empirical

7. Kearns and Singh (2002) dealt with discounted and undiscounted MDPs differently. In the discounted case the agent is required to halt after a polynomial amount of time and output a near-optimal policy from the current state, with high probability.

mean reward is

$$\hat{R}(s,a) := \frac{1}{n(s,a)} \sum_{i=1}^{n(s,a)} r[i].$$

Let $n(s,a,s')$ denote the number of times the agent has taken action a from state s and immediately transitioned to the state s'. Then, the *empirical transition distribution* is the distribution $\hat{T}(s,a)$ satisfying

$$\hat{T}(s'|s,a) := \frac{n(s,a,s')}{n(s,a)} \text{ for each } s' \in S.$$

In the R-MAX algorithm, the action-selection step is always to choose the action that maximizes the current action value, $Q(s,\cdot)$. The update step is to solve the following set of Bellman equations:

$$\begin{aligned} Q(s,a) &= \hat{R}(s,a) + \gamma \sum_{s'} \hat{T}(s'|s,a) \max_{a'} Q(s',a'), && \text{if } n(s,a) \geq m, \\ Q(s,a) &= U(s,a), && \text{otherwise,} \end{aligned} \tag{4}$$

where $\hat{R}(s,a)$ and $\hat{T}(\cdot|s,a)$ are the empirical (maximum-likelihood) estimates for the reward and transition distribution of state-action pair (s,a) using only data from the first m observations of (s,a). Solving this set of equations is equivalent to computing the optimal action-value function of an MDP, which we call *Model(R-MAX)*. This MDP uses the empirical transition and reward distributions for those state-action pairs that have been experienced by the agent at least m times. Rather than attempt to model the other state-action pairs, we assert their value to be $U(s,a)$, which is guaranteed to be an upper bound on the true value function. An important point is that R-MAX uses *only the first m samples* in the empirical model. To avoid complicated notation, we redefine $n(s,a)$ to be the minimum of m and the number of times state-action pair (s,a) has been experienced. This usage is consistent with the pseudo-code provided in Algorithm 1. That is, the computation of $\hat{R}(s,a)$ and $\hat{T}(s'|s,a)$ in Equation 4, uses only the first $n(s,a) = m$ samples.

Any implementation of R-MAX must choose a technique for solving the set of Equations 4 such as dynamic programming and linear programming approaches (Puterman, 1994), and this choice will affect the computational complexity of the algorithm. However, for concreteness we choose *value iteration* (Puterman, 1994), a relatively simple and fast MDP solving routine that is widely used in practice. Rather than require exact solution of Equations 4, a more practical approach is to only guarantee a near-optimal greedy policy. The following two classic results are useful in quantifying the number of iterations needed.

Proposition 3 *(Corollary 2 from Singh and Yee 1994) Let $Q'(\cdot,\cdot)$ and $Q^*(\cdot,\cdot)$ be two action-value functions over the same state and action spaces. Suppose that Q^* is the optimal value function of some MDP M. Let π be the greedy policy with respect to Q' and π^* be the greedy policy with respect to Q^*, which is the optimal policy for M. For any $\alpha > 0$ and discount factor $\gamma < 1$, if $\max_{s,a} \{|Q'(s,a) - Q^*(s,a)|\} \leq \alpha(1-\gamma)/2$, then $\max_s \{V^{\pi^*}(s) - V^{\pi}(s)\} \leq \alpha$.*

Proposition 4 *Let $\beta > 0$ be any real number satisfying $\beta < 1/(1-\gamma)$ where $\gamma < 1$ is the discount factor. Suppose that value iteration is run for $\left\lceil \frac{\ln(1/(\beta(1-\gamma)))}{1-\gamma} \right\rceil$ iterations where each initial action-value estimate, $Q(\cdot,\cdot)$, is initialized to some value between 0 and $1/(1-\gamma)$. Let $Q'(\cdot,\cdot)$ be the resulting action-value estimates. Then, we have that $\max_{s,a} \{|Q'(s,a) - Q^*(s,a)|\} \leq \beta$.*

Proof Let $Q_i(s,a)$ denote the action-value estimates after the i^{th} iteration of value iteration.[8] Let $\Delta_i := \max_{(s,a)} |Q^*(s,a) - Q_i(s,a)|$. Now, we have that

$$\begin{aligned}
\Delta_i &= \max_{(s,a)} |(R(s,a) + \gamma \sum_{s'} T(s,a,s')V^*(s')) - (R(s,a) + \gamma \sum_{s'} T(s,a,s')V_{i-1}(s'))| \\
&= \max_{(s,a)} |\gamma \sum_{s'} T(s,a,s')(V^*(s') - V_{i-1}(s'))| \\
&\leq \gamma \Delta_{i-1}.
\end{aligned}$$

Using this bound along with the fact that $\Delta_0 \leq 1/(1-\gamma)$ shows that $\Delta_i \leq \gamma^i/(1-\gamma)$. Setting this value to be at most β and solving for i yields $i \geq \frac{\ln(\beta(1-\gamma))}{\ln\gamma}$. We claim that

$$\frac{\ln \frac{1}{\beta(1-\gamma)}}{1-\gamma} \geq \frac{\ln(\beta(1-\gamma))}{\ln(\gamma)}. \tag{5}$$

Note that Equation 5 is equivalent to the statement $1-\gamma \leq -\ln\gamma$, which follows from the identity $e^x \geq 1+x$. ■

The previous two propositions imply that if we require value iteration to produce an α-optimal policy it is sufficient to run it for $O\left(\frac{\ln(1/(\alpha(1-\gamma)))}{1-\gamma}\right)$ iterations. The resulting pseudo-code for R-MAX is given in Algorithm 1. We have added a real-valued parameter, ε_1, that specifies the desired closeness to optimality of the policies produced by value iteration. In Section 3.2.2, we show that both m and ε_1 can be set as functions of the other input parameters, ε, δ, S, A, and γ, in order to make theoretical guarantees about the learning efficiency of R-MAX.

2.2 Delayed Q-learning

The *Delayed Q-learning* algorithm was introduced by Strehl et al. (2006b) as the first algorithm that is known to be PAC-MDP and its per-timestep computational demands are minimal (roughly equivalent to those of Q-learning). Due to its low memory requirements, it can also be viewed as a *model-free* algorithm and the first to be provably PAC-MDP. Its analysis is also noteworthy because the polynomial upper bound on its sample complexity is a significant improvement, asymptotically, over the best previously known upper bound for any algorithm, when only the dependence on S and A is considered.

The algorithm is called "delayed" because it waits until a state-action pair has been experienced m times before updating that state-action pair's associated action value, where m is a parameter provided as input. When it does update an action value, the update can be viewed as an average of the target values for the m most recently missed update opportunities. An important observation is that, when m is large enough, a Delayed Q-learning update will be sufficiently close to a true Bellman update (Lemma 22). In this sense, this algorithm is similar to Real-Time Dynamic Programming (Barto et al., 1995), but uses online transitions to dynamically form an approximate Bellman backup.

To encourage exploration, Delayed Q-learning uses the "optimism in the face of uncertainty" principle as in R-MAX. Specifically, its initial action-value function is an over-estimate of the true

8. The initial values are therefore denoted by $Q_0(\cdot,\cdot)$.

Algorithm 1 R-MAX

Inputs: $S, A, \gamma, m, \varepsilon_1$, and $U(\cdot,\cdot)$
for all (s,a) **do**
  $Q(s,a) \leftarrow U(s,a)$ // *action-value estimates*
  $r(s,a) \leftarrow 0$
  $n(s,a) \leftarrow 0$
  for all $s' \in S$ **do**
    $n(s,a,s') \leftarrow 0$
  end for
end for
for $t = 1,2,3,\cdots$ **do**
  Let s denote the state at time t.
  Choose action $a := \text{argmax}_{a' \in A} Q(s,a')$.
  Let r be the immediate reward and s' the next state after executing action a from state s.
  if $n(s,a) < m$ **then**
    $n(s,a) \leftarrow n(s,a) + 1$
    $r(s,a) \leftarrow r(s,a) + r$ // *Record immediate reward*
    $n(s,a,s') \leftarrow n(s,a,s') + 1$ // *Record immediate next-state*
    if $n(s,a) = m$ **then**
      for $i = 1,2,3,\cdots, \left\lceil \frac{\ln(1/(\varepsilon_1(1-\gamma)))}{1-\gamma} \right\rceil$ **do**
        for all $(\bar{s},\bar{a})$ **do**
          if $n(\bar{s},\bar{a}) \geq m$ **then**
            $Q(\bar{s},\bar{a}) \leftarrow \hat{R}(\bar{s},\bar{a}) + \gamma \sum_{s'} \hat{T}(s'|\bar{s},\bar{a}) \max_{a'} Q(s',a')$.
          end if
        end for
      end for
    end if
  end if
end for

function; during execution, the successive value function estimates remain over-estimates with high probability, thanks to the delayed update rule (Lemma 23).

Like R-MAX, Delayed Q-learning performs a finite number of action-value updates. Due to the strict restrictions on the computational demands used by Delayed Q-learning, slightly more sophisticated internal logic is needed to guarantee this property. Pseudo-code[9] for Delayed Q-learning is provided in Algorithm 2. More details are provided in the following subsections.

In addition to the standard inputs, the algorithm also relies on two free parameters,

- $\varepsilon_1 \in (0,1)$: Used to provide a constant "exploration bonus" that is added to each action-value estimate when it is updated.

9. Compared to the implementation provided by Strehl et al. (2006b), we have modified the algorithm to keep track of $b(s,a)$, the "beginning" timestep for the current attempted update for (s,a). The original pseudo-code kept track of $t(s,a)$, the time of the last attempted update for (s,a). The original implementation is less efficient and adds a factor of 2 to the computational bounds. The analysis of Strehl et al. (2006b) also applies to the pseudo-code presented here, however.

Algorithm 2 Delayed Q-learning

Inputs: $S, A, \gamma, m, \varepsilon_1$, and $U(\cdot,\cdot)$
for all (s,a) **do**
  $Q(s,a) \leftarrow U(s,a)$ *// action-value estimates*
  $AU(s,a) \leftarrow 0$ *// used for attempted updates*
  $l(s,a) \leftarrow 0$ *// counters*
  $b(s,a) \leftarrow 0$ *// beginning timestep of attempted update*
  $LEARN(s,a) \leftarrow true$ *// the LEARN flags*
end for
$t^* \leftarrow 0$ *// time of most recent action value change*
for $t = 1,2,3,\cdots$ **do**
  Let s denote the state at time t.
  Choose action $a := \text{argmax}_{a' \in A} Q(s,a')$.
  Let r be the immediate reward and s' the next state after executing action a from state s.
  if $b(s,a) \leq t^*$ **then**
    $LEARN(s,a) \leftarrow true$
  end if
  if $LEARN(s,a) = true$ **then**
    if $l(s,a) = 0$ **then**
      $b(s,a) \leftarrow t$
    end if
    $l(s,a) \leftarrow l(s,a) + 1$
    $AU(s,a) \leftarrow AU(s,a) + r + \gamma \max_{a'} Q(s',a')$
    if $l(s,a) = m$ **then**
      if $Q(s,a) - AU(s,a)/m \geq 2\varepsilon_1$ **then**
        $Q(s,a) \leftarrow AU(s,a)/m + \varepsilon_1$
        $t^* \leftarrow t$
      else if $b(s,a) > t^*$ **then**
        $LEARN(s,a) \leftarrow false$
      end if
      $AU(s,a) \leftarrow 0$
      $l(s,a) \leftarrow 0$
    end if
  end if
end for

- A positive integer m: Represents the number of experiences of a state-action pair before an update is allowed.

In the analysis of Section 3.3, we provide precise values for m and ε_1 in terms of the other inputs (S, A, ε, δ, and γ) that guarantee the resulting algorithm is PAC-MDP. In addition to its action-value estimates, $Q(s,a)$, the algorithm also maintains the following internal variables,

- $l(s,a)$ for each (s,a): The number of samples (or target values) gathered for (s,a).

- $AU(s,a)$ for each (s,a): Stores the running sum of target values used to update $Q(s,a)$ once enough samples have been gathered.
- $b(s,a)$ for each (s,a): The timestep for which the first experience of (s,a) was obtained for the most recent or ongoing attempted update.
- $LEARN(s,a) \in \{true, false\}$ for each (s,a): A Boolean flag that indicates whether or not, samples are being gathered for (s,a).

2.2.1 THE UPDATE RULE

Suppose that, at time $t \geq 1$, action a is performed from state s, resulting in an *attempted update*, according to the rules to be defined in Section 2.2.2. Let $s_{k_1}, s_{k_2}, \ldots, s_{k_m}$ be the m most recent next-states observed from executing (s,a) at times $k_1 < k_2 < \cdots < k_m$, respectively ($k_m = t$). For the remainder of the paper, we also let r_i denote the i^{th} reward received during the execution of Delayed Q-learning.

Thus, at time k_i, action a was taken from state s, resulting in a transition to state s_{k_i} and an immediate reward r_{k_i}. After the t^{th} action, the following update occurs:

$$Q_{t+1}(s,a) = \frac{1}{m} \sum_{i=1}^{m} (r_{k_i} + \gamma V_{k_i}(s_{k_i})) + \varepsilon_1 \tag{6}$$

as long as performing the update would result in a new action-value estimate that is at least ε_1 smaller than the previous estimate. In other words, the following equation must be satisfied for an update to occur:

$$Q_t(s,a) - \left(\frac{1}{m} \sum_{i=1}^{m} (r_{k_i} + \gamma V_{k_i}(s_{k_i})) \right) \geq 2\varepsilon_1. \tag{7}$$

If this condition does not hold, then no update is performed, and so $Q_{t+1}(s,a) = Q_t(s,a)$.

2.2.2 MAINTENANCE OF THE *LEARN* FLAGS

We provide an intuition behind the behavior of the *LEARN* flags. Please see Algorithm 2 for a formal description of the update rules. The main computation of the algorithm is that every time a state-action pair (s,a) is experienced m times, an update of $Q(s,a)$ is attempted as in Section 2.2.1. For our analysis to hold, however, we cannot allow an infinite number of attempted updates. Therefore, attempted updates are only allowed for (s,a) when $LEARN(s,a)$ is *true*. Besides being set to *true* initially, $LEARN(s,a)$ is also set to *true* when any state-action pair is updated (because our estimate $Q(s,a)$ may need to reflect this change). $LEARN(s,a)$ can only change from *true* to *false* when no updates are made during a length of time for which (s,a) is experienced m times and the next attempted update of (s,a) fails. In this case, no more attempted updates of (s,a) are allowed until another action-value estimate is updated.

2.2.3 DELAYED Q-LEARNING'S MODEL

Delayed Q-learning was introduced as a *model-free* algorithm. This terminology was justified by noting that the space complexity of Delayed Q-learning, which is $O(SA)$, is much less than what is needed in the worst case to completely represent an MDP's transition probabilities ($O(S^2A)$).

However, there is a sense in which Delayed Q-learning can be thought of as using a model. This interpretation follows from the fact that Delayed Q-learning's update (Equation 6) is identical to ε_1 plus the result of a full Bellman backup using the empirical (maximum likelihood) model derived from the m most recent experiences of the state-action pair being updated. Since m is much less than what is needed to accurately model the true transition probability (in the $L1$ distance metric), we say that Delayed Q-learning uses a *sparse model* (Kearns and Singh, 1999). In fact, Delayed Q-learning uses this sparse model precisely once, throws it away, and then proceeds to gather experience for another sparse model. When $m = 1$, this process may occur on every timestep and the algorithm behaves very similarly to a version of Q-learning that uses a unit learning rate.

3. PAC-MDP Analysis

First, we present a general framework that allows us to prove the bounds for both algorithms. We then proceed to analyze R-MAX and Delayed Q-learning.

3.1 General Framework

We now develop some theoretical machinery to prove PAC-MDP statements about various algorithms. Our theory will be focused on algorithms that maintain a table of action values, $Q(s,a)$, for each state-action pair (denoted $Q_t(s,a)$ at time t).[10] We also assume an algorithm always chooses actions greedily with respect to the action values. This constraint is not really a restriction, since we could define an algorithm's action values as 1 for the action it chooses and 0 for all other actions. However, the general framework is understood and developed more easily under the above assumptions. For convenience, we also introduce the notation $V(s)$ to denote $\max_a Q(s,a)$ and $V_t(s)$ to denote $V(s)$ at time t.

Definition 5 *Suppose an RL algorithm $\mathcal{A}$ maintains a value, denoted $Q(s,a)$, for each state-action pair $(s,a) \in \mathcal{S} \times \mathsf{A}$. Let $Q_t(s,a)$ denote the estimate for (s,a) immediately before the t^{th} action of the agent. We say that $\mathcal{A}$ is a* **greedy algorithm** *if the t^{th} action of $\mathcal{A}$, a_t, is $a_t := \text{argmax}_{a \in \mathsf{A}} Q_t(s_t,a)$, where s_t is the t^{th} state reached by the agent.*

For all algorithms, the action values $Q(\cdot,\cdot)$ are implicitly maintained in separate max-priority queues (implemented with max-heaps, say) for each state. Specifically, if $\mathsf{A} = \{a_1, \ldots, a_k\}$ is the set of actions, then for each state s, the values $Q(s,a_1), \ldots, Q(s,a_k)$ are stored in a single priority queue. Therefore, the operations $\max_{a' \in \mathsf{A}} Q(s,a)$ and $\text{argmax}_{a' \in \mathsf{A}} Q(s,a)$, which appear in almost every algorithm, takes constant time, but the operation $Q(s,a) \leftarrow V$ for any value V takes $O(\ln(A))$ time (Cormen et al., 1990). It is possible that other data structures may result in faster algorithms.

The following is a definition of a new MDP that will be useful in our analysis.

Definition 6 *Let $M = \langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$ be an MDP with a given set of action values, $Q(s,a)$, for each state-action pair (s,a), and a set K of state-action pairs, called the* **known state-action pairs**. *We define the* **known state-action MDP** *$M_K = \langle \mathcal{S} \cup \{z_{s,a} | (s,a) \notin K\}, \mathsf{A}, T_K, R_K, \gamma \rangle$ as follows. For each unknown state-action pair, $(s,a) \notin K$, we add a new state $z_{s,a}$ to M_K, which has self-loops for each*

10. However, the main result in this subsection (Theorem 10) does not rely on the algorithm having an explicit representation of each action value. For example, they could be implicitly held inside of a function approximator (e.g., Brunskill et al. 2008).

action ($T_K(z_{s,a}|z_{s,a},\cdot) = 1$). For all $(s,a) \in K$, $R_K(s,a) = R(s,a)$ and $T_K(\cdot|s,a) = T(\cdot|s,a)$. For all $(s,a) \notin K$, $R_K(s,a) = Q(s,a)(1-\gamma)$ and $T_K(z_{s,a}|s,a) = 1$. For the new states, the reward is $R_K(z_{s,a},\cdot) = Q(s,a)(1-\gamma)$.

The known state-action MDP is a generalization of the standard notions of a "known state MDP" of Kearns and Singh (2002) and Kakade (2003). It is an MDP whose dynamics (reward and transition functions) are equal to the true dynamics of M for a subset of the state-action pairs (specifically those in K). For all other state-action pairs, the value of taking those state-action pairs in M_K (and following any policy from that point on) is equal to the current action-value estimates $Q(s,a)$. We intuitively view K as a set of state-action pairs for which the agent has sufficiently accurate estimates of their dynamics.

Definition 7 *For algorithm $\mathcal{A}$, for each timestep t, let K_t (we drop the subscript t if t is clear from context) be a set of state-action pairs defined arbitrarily in a way that depends only on the history of the agent up to timestep t (before the $(t)^{\text{th}}$ action). We define A_K to be the event, called the* **escape event**, *that some state-action pair $(s,a) \notin K_t$ is experienced by the agent at time t.*

The following is a well-known result of the Chernoff-Hoeffding Bound and will be needed later; see Li (2009, Lemma 56) for a slightly improved result.

Lemma 8 *Suppose a weighted coin, when flipped, has probability $p > 0$ of landing with heads up. Then, for any positive integer k and real number $\delta \in (0,1)$, there exists a number $m = O((k/p)\ln(1/\delta))$, such that after m tosses, with probability at least $1-\delta$, we will observe k or more heads.*

One more technical lemma is needed before presenting the main result in this section. Note that even if we assume $V^*_M(s) \le V_{\max}$ and $Q(s,a) \le V_{\max}$ for all $s \in \mathcal{S}$ and $a \in \mathsf{A}$, it may not be true that $V^*_{M_K}(s) \le V_{\max}$. However, the following lemma shows we may instead use $2V_{\max}$ as an upper bound.

Lemma 9 *Let $M = \langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$ be an MDP whose optimal value function is upper bounded by $V_{\max}$. Furthermore, let M_K be a known state-action MDP for some $K \subseteq \mathcal{S} \times \mathsf{A}$ defined using value function $Q(s,a)$. Then, $V^*_{M_K}(s) \le V_{\max} + \max_{s',a'} Q(s',a')$ for all $s \in \mathcal{S}$.*

Proof For any policy π and any state $s \in \mathcal{S}$, let $(s_1,a_1,r_1,s_2,a_2,r_2,s_3,a_3,r_3,\ldots)$ be a path generated by starting in state $s = s_1$ and following π in the known state-action MDP, M_K, where s_t and r_t are the state and reward at timestep t, and $a_t = \pi(s_t)$ for all t. The value function, $V^\pi_{M_K}(s)$, can be written as (see, e.g., Sutton and Barto 1998)

$$V^\pi_{M_K}(s) = \mathbf{E}_{M_K}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \mid s_1 = s, \pi\right],$$

which says the quantity $V^\pi_{M_K}(s)$ is the expected discounted total reward accumulated on this random path. Here, we use $\mathbf{E}_{M_K}$ to denote the expectation with respect to randomness in the MDP M_K.

Denote by τ be the *first* timestep in which $(s_\tau, a_\tau) \notin K$; note $\tau = \infty$ if all visited state-actions are in K. Due to construction of M_K, if τ is finite, then

$$\begin{aligned} s_\tau &= s_{\tau+1} = s_{\tau+2} = \cdots \\ a_\tau &= a_{\tau+1} = a_{\tau+2} = \cdots = \pi(s_\tau) \\ r_\tau &= r_{\tau+1} = r_{\tau+2} = \cdots = (1-\gamma)Q(s_\tau, a_\tau). \end{aligned}$$

Thus, for any fixed $\tau \geq 1$, the discounted total reward

$$\begin{aligned} & r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \\ &= \; r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1} + \gamma^{\tau-1} Q(s_\tau, a_\tau) \\ &\leq \; r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1} + \max_{s',a'} Q(s',a'), \end{aligned}$$

where the first step is due to the way we define transition/reward functions in M_K for state-actions outside K. The above upper bound holds for all fixed value of τ (finite or infinite), and so

$$\begin{aligned} & \mathbf{E}_{M_K}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \mid s_1 = s, \pi\right] \\ &\leq \; \mathbf{E}_{M_K}\left[r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1} \mid s_1 = s, \pi\right] + \max_{s',a'} Q(s',a'). \end{aligned}$$

Finally, since the transition and reward functions of M and M_K are identical for state-actions in K, we have

$$\begin{aligned} & \mathbf{E}_{M_K}\left[r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1} \mid s_1 = s, \pi\right] \\ &= \; \mathbf{E}_{M}\left[r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1} \mid s_1 = s, \pi\right], \end{aligned}$$

which implies

$$\begin{aligned} & \mathbf{E}_{M_K}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \mid s_1 = s, \pi\right] \\ &\leq \; \mathbf{E}_{M}\left[r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1}\right] + \max_{s',a'} Q(s',a') \\ &\leq \; V_M^{\pi}(s) + \max_{s',a'} Q(s',a') \\ &\leq \; V_{\max} + \max_{s',a'} Q(s',a'). \end{aligned}$$

■

Note that all learning algorithms we consider take ε and δ as input. We let $\mathcal{A}(\varepsilon,\delta)$ denote the version of algorithm $\mathcal{A}$ parameterized with ε and δ. The proof of Theorem 10 follows the structure of the work of Kakade (2003), but generalizes several key steps. The theorem also generalizes a previous result by Strehl et al. (2006a) by taking the admissible heuristic into account.

Theorem 10 *Let $\mathcal{A}(\varepsilon,\delta)$ be any greedy learning algorithm such that, for every timestep t, there exists a set K_t of state-action pairs that depends only on the agent's history up to timestep t. We assume that $K_t = K_{t+1}$ unless, during timestep t, an update to some state-action value occurs or the escape event A_K happens. Let M_{K_t} be the known state-action MDP and π_t be the current greedy policy, that is, for all states s, $\pi_t(s) = \operatorname{argmax}_a Q_t(s,a)$. Furthermore, assume $Q_t(s,a) \leq V_{\max}$ for all t and (s,a). Suppose that for any inputs ε and δ, with probability at least $1-\delta$, the following conditions hold for all states s, actions a, and timesteps t: (1) $V_t(s) \geq V^*(s) - \varepsilon$ (optimism), (2) $V_t(s) - V_{M_{K_t}}^{\pi_t}(s) \leq \varepsilon$ (accuracy), and (3) the total number of updates of action-value estimates plus the number of times the escape event from K_t, A_K, can occur is bounded by $\zeta(\varepsilon,\delta)$ (learning complexity). Then, when $\mathcal{A}(\varepsilon,\delta)$ is executed on any MDP M, it will follow a 4ε-optimal policy from its current state on all but*

$$O\left(\frac{V_{\max}\zeta(\varepsilon,\delta)}{\varepsilon(1-\gamma)} \ln\frac{1}{\delta} \ln\frac{1}{\varepsilon(1-\gamma)}\right)$$

timesteps, with probability at least $1-2\delta$.

Proof Suppose that the learning algorithm $\mathcal{A}(\varepsilon,\delta)$ is executed on MDP M. Fix the history of the agent up to the t^{th} timestep and let s_t be the t^{th} state reached. Let $\mathcal{A}_t$ denote the current (non-stationary) policy of the agent. Let $H = \frac{1}{1-\gamma}\ln\frac{1}{\varepsilon(1-\gamma)}$. From Lemma 2 of Kearns and Singh (2002), we have that $|V^{\pi}_{M_{K_t}}(s,H) - V^{\pi}_{M_{K_t}}(s)| \leq \varepsilon$, for any state s and policy π. Let W denote the event that, after executing policy $\mathcal{A}_t$ from state s_t in M for H timesteps, one of the two following events occur: (a) the algorithm performs a successful update (a change to any of its action values) of some state-action pair (s,a), or (b) some state-action pair $(s,a) \notin K_t$ is experienced (escape event A_K). Assuming the three conditions in the theorem statement hold, we have the following:

$$\begin{aligned}
&V^{\mathcal{A}_t}_M(s_t,H) \\
&\quad\geq\; V^{\pi_t}_{M_{K_t}}(s_t,H) - 2V_{\max}\Pr(W) \\
&\quad\geq\; V^{\pi_t}_{M_{K_t}}(s_t) - \varepsilon - 2V_{\max}\Pr(W) \\
&\quad\geq\; V(s_t) - 2\varepsilon - 2V_{\max}\Pr(W) \\
&\quad\geq\; V^*(s_t) - 3\varepsilon - 2V_{\max}\Pr(W).
\end{aligned}$$

The first step above follows from the fact that following $\mathcal{A}_t$ in MDP M results in behavior identical to that of following π_t in M_{K_t} unless event W occurs, in which case a loss of at most $2V_{\max}$ can occur (Lemma 9). The second step follows from the definition of H above. The third and final steps follow from Conditions 2 and 1, respectively, of the proposition.

Now, suppose that $\Pr(W) < \frac{\varepsilon}{2V_{\max}}$. Then, we have that the agent's policy on timestep t is 4ε-optimal:

$$V^{\mathcal{A}_t}_M(s_t) \geq V^{\mathcal{A}_t}_M(s_t,H) \geq V^*_M(s_t) - 4\varepsilon.$$

Otherwise, we have that $\Pr(W) \geq \frac{\varepsilon}{2V_{\max}}$, which implies that an agent following $\mathcal{A}_t$ will either perform a successful update in H timesteps, or encounter some $(s,a) \notin K_t$ in H timesteps, with probability at least $\frac{\varepsilon}{2V_{\max}}$. Call such an event a "success". Then, by Lemma 8, after $O(\frac{\zeta(\varepsilon,\delta)HV_{\max}}{\varepsilon}\ln 1/\delta)$ timesteps t where $\Pr(W) \geq \frac{\varepsilon}{2V_{\max}}$, $\zeta(\varepsilon,\delta)$ successes will occur, with probability at least $1-\delta$. Here, we have identified the event that a success occurs after following the agent's policy for H steps with the event that a coin lands with heads facing up. However, by Condition 3 of the proposition, with probability at least $1-\delta$, $\zeta(\varepsilon,\delta)$ is the maximum number of successes that will occur throughout the execution of the algorithm.

To summarize, we have shown that with probability $1-2\delta$, the agent will execute a 4ε-optimal policy on all but $O(\frac{\zeta(\varepsilon,\delta)HV_{\max}}{\varepsilon}\ln\frac{1}{\delta}) = O(\frac{\zeta(\varepsilon,\delta)V_{\max}}{\varepsilon(1-\gamma)}\ln\frac{1}{\delta}\ln\frac{1}{\varepsilon(1-\gamma)})$ timesteps. ■

3.2 Analysis of R-MAX

We will analyze R-MAX using the tools from Section 3.1.

3.2.1 COMPUTATIONAL COMPLEXITY

When the initial value function is $U(s,a) = 1/(1-\gamma)$ for all (s,a), there is a simple way to change the R-MAX algorithm that has a minimal affect on its behavior and saves greatly on computation. The important observation is that for a fixed state s, the maximum action-value estimate, $\max_a Q(s,a)$ will be $1/(1-\gamma)$ until all actions have been tried m times. Thus, there is no need to

run value iteration (lines 17 to 25 in Algorithm 1) until each action has been tried exactly m times. In addition, if there are some actions that have been tried m times and others that have not, the algorithm should choose one of the latter. One method to accomplish this balance is to order each action and try one after another until all are chosen m times. Kearns and Singh (2002) called this behavior "balanced wandering". However, it is not necessary to use balanced wandering; for example, it would be perfectly fine to try the first action m times, the second action m times, and so on. Any deterministic method for breaking ties in line 11 of Algorithm 1 is valid as long as mA experiences of a state-action pair results in all action being chosen m times.

On most timesteps, the R-MAX algorithm performs a constant amount of computation to choose its next action. Only when a state's last action has been tried m times does it solve its internal model. Our version of R-MAX uses value iteration to solve its model. Therefore, the per-timestep computational complexity of R-MAX is

$$\Theta\left(SA(S+\ln(A))\left(\frac{1}{1-\gamma}\right)\ln\frac{1}{\varepsilon_1(1-\gamma)}\right).$$

This expression is derived using the fact that value iteration performs $\left\lceil \frac{1}{1-\gamma}\ln\frac{1}{\varepsilon_1(1-\gamma)}\right\rceil$ iterations, where each iteration involves SA full Bellman backups (one for each state-action pair). A Bellman backup requires examining all possible $O(S)$ successor states and the update to the priority queue takes time $O(\ln(A))$. Note that R-MAX updates its model at most S times. From this observation we see that the total computation time of R-MAX is $O\left(B+\frac{S^2A(S+\ln(A))}{1-\gamma}\ln\frac{1}{\varepsilon_1(1-\gamma)}\right)$, where B is the number of timesteps for which R-MAX is executed.

When a general admissible initial value function U is used, we need to run value iteration whenever some $n(s,a)$ reaches the threshold m. In this case, a similar analysis shows that the total computation time of R-MAX is $O\left(B+\frac{S^2A^2(S+\ln(A))}{1-\gamma}\ln\frac{1}{\varepsilon_1(1-\gamma)}\right)$.

3.2.2 SAMPLE COMPLEXITY

The main result of this section is the following theorem.

Theorem 11 *Suppose that* $0\le\varepsilon<\frac{1}{1-\gamma}$ *and* $0\le\delta<1$ *are two real numbers and* $M=\langle\mathcal{S},\mathsf{A},T,\mathcal{R},\gamma\rangle$ *is any MDP. There exists inputs* $m=m(\frac{1}{\varepsilon},\frac{1}{\delta})$ *and* ε_1*, satisfying* $m(\frac{1}{\varepsilon},\frac{1}{\delta})=O\left(\frac{(S+\ln(SA/\delta))V_{\max}^2}{\varepsilon^2(1-\gamma)^2}\right)$ *and* $\frac{1}{\varepsilon_1}=O(\frac{1}{\varepsilon})$*, such that if R-MAX is executed on* M *with inputs* m *and* ε_1*, then the following holds. Let* $\mathcal{A}_t$ *denote R-MAX's policy at time* t *and* s_t *denote the state at time* t*. With probability at least* $1-\delta$*,* $V_M^{\mathcal{A}_t}(s_t)\ge V_M^*(s_t)-\varepsilon$ *is true for all but*

$$O\left(\frac{|\{(s,a)\in\mathcal{S}\times\mathsf{A}|U(s,a)\ge V^*(s)-\varepsilon\}|}{\varepsilon^3(1-\gamma)^3}\left(S+\ln\frac{SA}{\delta}\right)V_{\max}^3\ln\frac{1}{\delta}\ln\frac{1}{\varepsilon(1-\gamma)}\right)$$

timesteps t*.*

First, we discuss the accuracy of the model maintained by R-MAX. The following lemma shows that two MDPs with similar transition and reward functions have similar value functions. Thus, an agent need only ensure accuracy in the transitions and rewards of its model to guarantee near-optimal behavior.

Lemma 12 *(Strehl and Littman, 2005) Let $M_1 = \langle \mathcal{S}, \mathsf{A}, T_1, R_1, \gamma \rangle$ and $M_2 = \langle \mathcal{S}, \mathsf{A}, T_2, R_2, \gamma \rangle$ be two MDPs with non-negative rewards bounded by 1 and optimal value functions bounded by $V_{\max}$. Suppose that $|R_1(s,a) - R_2(s,a)| \leq \alpha$ and $\|T_1(s,a,\cdot) - T_2(s,a,\cdot)\|_1 \leq 2\beta$ for all states s and actions a. There exists a constant $C > 0$ such that for any $0 \leq \varepsilon \leq 1/(1-\gamma)$ and stationary policy π, if $\alpha = 2\beta = C\varepsilon(1-\gamma)/V_{\max}$, then*

$$|Q_1^\pi(s,a) - Q_2^\pi(s,a)| \leq \varepsilon.$$

Let $n_t(s,a)$ denote the value of $n(s,a)$ at time t during execution of the algorithm. For R-MAX, let the "known" state-action pairs K_t, at time t (See Definition 6), to be

$$K_t := \{(s,a) \in \mathcal{S} \times \mathsf{A} | n_t(s,a) \geq m\},$$

which is dependent on the parameter m that is provided as input to the algorithm. In other words, K_t is the set of state-action pairs that have been experienced by the agent at least m times. We will show that for large enough m, the dynamics, transition and reward, associated with these pairs can be accurately approximated by the agent.

The following event will be used in our proof that R-MAX is PAC-MDP. We will provide a sufficient condition (specifically, L_1-accurate transition and reward functions) to guarantee that the event occurs, with high probability. In words, the condition says that the value of any state s, under any policy, in the empirical known state-action MDP ($\hat{M}_{K_t}$) is ε_1-close to its value in the true known state-action MDP (M_{K_t}).

Event A1 *For all stationary policies π, timesteps t and states s during execution of the R-MAX algorithm on some MDP M, $|V^\pi_{M_{K_t}}(s) - V^\pi_{\hat{M}_{K_t}}(s)| \leq \varepsilon_1$.*

Next, we quantify the number of samples needed from both the transition and reward distributions for a state-action pair to compute accurate approximations.

Lemma 13 *Suppose that $r[1], r[2], \ldots, r[m]$ are m rewards drawn independently from the reward distribution, $\mathcal{R}(s,a)$, for state-action pair (s,a). Let $\hat{R}(s,a)$ be the empirical (maximum-likelihood) estimate of $\mathcal{R}(s,a)$. Let δ_R be any positive real number less than 1. Then, with probability at least $1 - \delta_R$, we have that $|\hat{R}(s,a) - \mathcal{R}(s,a)| \leq \varepsilon^R_{n(s,a)}$, where*

$$\varepsilon^R_m := \sqrt{\frac{\ln(2/\delta_R)}{2m}}.$$

Proof This result follows directly from Hoeffding's bound (Hoeffding, 1963). ■

Lemma 14 *Suppose that $\hat{T}(s,a)$ is the empirical transition distribution for state-action pair (s,a) using m samples of next states drawn independently from the true transition distribution $T(s,a)$. Let δ_T be any positive real number less than 1. Then, with probability at least $1 - \delta_T$, we have that $\|T(s,a) - \hat{T}(s,a)\|_1 \leq \varepsilon^T_{n(s,a)}$ where*

$$\varepsilon^T_m = \sqrt{\frac{2[\ln(2^S - 2) - \ln(\delta_T)]}{m}}.$$

Proof The result follows immediately from an application of Theorem 2.1 of Weissman et al. (2003).[11] ■

Lemma 15 *There exists a constant C such that if R-MAX with parameters m and ε_1 is executed on any MDP $M = \langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$ and m satisfies*

$$m \geq C V_{\max}^2 \left(\frac{S + \ln(SA/\delta)}{{\varepsilon_1}^2 (1-\gamma)^2} \right) = \tilde{O}\left(\frac{S V_{\max}^2}{{\varepsilon_1}^2 (1-\gamma)^2} \right),$$

then Event A1 will occur with probability at least $1 - \delta$.

Proof Event A1 occurs if R-MAX maintains a close approximation of its known state-action MDP. By Lemmas 9 and 12, it is sufficient to obtain $(C\varepsilon_1(1-\gamma)/V_{\max})$-approximate transition and reward functions (where C is a constant), for those state-action pairs in K_t. The transition and reward functions that R-MAX uses are the maximum-likelihood estimates, using only the first m samples (of immediate reward and next-state pairs) for each $(s,a) \in K$. Intuitively, as long as m is large enough, the empirical estimates for these state-action pairs will be accurate, with high probability.[12] Consider a fixed state-action pair (s,a). From Lemma 13, we can guarantee the empirical reward distribution is accurate enough, with probability at least $1 - \delta'$, as long as $\sqrt{\frac{\ln(2/\delta')}{2m}} \leq C\varepsilon_1(1-\gamma)/V_{\max}$. From Lemma 14, we can guarantee the empirical transition distribution is accurate enough, with probability at least $1 - \delta'$, as long as $\sqrt{\frac{2[\ln(2^S - 2) - \ln(\delta')]}{m}} \leq C\varepsilon_1(1-\gamma)/V_{\max}$. It is possible to choose m, as a function of the parameters of the MDP M, large enough so that both these expressions are satisfied but small enough so that

$$m \propto \frac{S + \ln(1/\delta')}{\varepsilon_1^2 (1-\gamma)^2} V_{\max}^2.$$

With this choice, we guarantee that the empirical reward and empirical distribution for a single state-action pair will be sufficiently accurate, with high probability. However, to apply the simulation bounds of Lemma 12, we require accuracy for all state-action pairs. To ensure a total failure probability of δ, we set $\delta' = \delta/(2SA)$ in the above equations and apply the union bound over all state-action pairs. ■

Proof (of Theorem 11). We apply Theorem 10. Let $\varepsilon_1 = \varepsilon/2$. Assume that Event A1 occurs. Consider some fixed time t. First, we verify Condition 1 of the theorem. We have that $V_t(s) \geq V^*_{\hat{M}_{K_t}}(s) - \varepsilon_1 \geq V^*_{M_{K_t}}(s) - 2\varepsilon_1 \geq V^*(s) - 2\varepsilon_1$. The first inequality follows from the fact that

11. The result of Weissman et al. (2003) is established using an information-theoretic argument. A similar result can be obtained (Kakade, 2003) by the multiplicative form of Chernoff's bounds.

12. There is a minor technicality here. The samples, in the form of immediate rewards and next states, experienced by an online agent in an MDP are not necessarily independent samples. The reason is that the learning environment or the agent could prevent future experiences of state-action pairs based on previously observed outcomes. Nevertheless, all the tail inequality bounds, including the Chernoff and Hoeffding Bounds, that hold for independent samples also hold for online samples in MDPs that can be viewed as martingales, a fact that follows from the Markov property. There is an extended discussion and formal proof of this fact elsewhere (Strehl and Littman, 2008b). An excellent review (with proofs) of the tail inequalities for martingales that we use in the present paper is by McDiarmid (1989).

R-MAX computes its action values by computing an ε_1-approximate solution of its internal model ($\hat{M}_{K_t}$) (using Proposition 4). The second inequality follows from Event A1 and the third from the fact that M_{K_t} can be obtained from M by removing certain states and replacing them with a maximally rewarding state whose actions are self-loops, an operation that only increases the value of any state. Next, we note that Condition 2 of the theorem follows from Event A1. Finally, observe that the learning complexity, $\zeta(\varepsilon,\delta) \leq |\{(s,a)|U(s,a) \geq V^*(s)-\varepsilon\}|m$. To see this fact, first note that state-action pair (s,a) with $U(s,a) < V^*(s)-\varepsilon$ will never be experienced, with high probability, because initially the agent chooses actions greedily with respect to $U(s,a)$ and there always exists another action a' such that $Q_t(s,a') > V^*(s)-\varepsilon$. Next, note that each time an escape occurs, some $(s,a) \notin K$ is experienced. However, once (s,a) is experienced m times, it becomes part of and never leaves the set K. To guarantee that Event A1 occurs with probability at least $1-\delta$, we use Lemma 15 to set m. ■

3.3 Analysis of Delayed Q-learning

In this section, we analyze the computational and sample complexity of Delayed Q-learning.

3.3.1 COMPUTATIONAL COMPLEXITY

On most timesteps, Delayed Q-learning performs only a constant amount of computation. Its worst-case computational complexity per timestep is

$$\Theta(\ln(A)),$$

where the logarithmic term is due to updating the priority queue that holds the action-value estimates for the current state. Since Delayed Q-learning performs at most $SA\left(1+\frac{SA}{(1-\gamma)\varepsilon_1}\right)$ attempted updates (see Lemma 19), each update involves m transitions, and each transition requires computing the greedy action whose computation complexity is $O(\ln(A))$, the total computation time of Delayed Q-learning is

$$O\left(B+\frac{mS^2A^2\ln(A)}{\varepsilon_1(1-\gamma)}\right),$$

where B is the number of timesteps for which Delayed Q-learning is executed. Since the number of attempted updates is bounded by a constant, the amortized computation time per step is $O(1)$ as B approaches ∞.

3.3.2 SAMPLE COMPLEXITY

In this section, we show that Delayed Q-learning is PAC-MDP.

Theorem 16 *(Strehl et al., 2006b) Suppose that* $0 \leq \varepsilon < \frac{1}{1-\gamma}$ *and* $0 \leq \delta < 1$ *are two real numbers and* $M = \langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$ *is any MDP. There exists inputs* $m = m(\frac{1}{\varepsilon}, \frac{1}{\delta})$ *and* ε_1*, satisfying* $m(\frac{1}{\varepsilon}, \frac{1}{\delta}) = O\left(\frac{(1+\gamma V_{\max})^2}{\varepsilon_1^2}\ln\frac{SA}{\varepsilon_1\delta(1-\gamma)}\right)$ *and* $\frac{1}{\varepsilon_1} = O(\frac{1}{\varepsilon(1-\gamma)})$*, such that if Delayed Q-learning is executed on* M*, then the following holds. Let* $\mathcal{A}_t$ *denote Delayed Q-learning's policy at time* t *and* s_t *denote the state at*

time t. With probability at least $1-\delta$, $V_M^{\mathcal{A}_t}(s_t) \geq V_M^*(s_t) - \varepsilon$ is true for all but

$$O\left(\frac{V_{\max}(1+\gamma V_{\max})^2 \sum_{(s,a)\in\mathcal{S}\times\mathsf{A}}[U(s,a)-V^*(s)]_+}{\varepsilon^4(1-\gamma)^4} \ln\frac{1}{\delta} \ln\frac{1}{\varepsilon(1-\gamma)} \ln\frac{SA}{\delta\varepsilon(1-\gamma)}\right)$$

timesteps t.

Definition 17 *An* **update** *(or* **successful update***) of state-action pair (s,a) is a timestep t for which a change to the action-value estimate $Q(s,a)$ occurs. An* **attempted update** *of state-action pair (s,a) is a timestep t for which (s,a) is experienced, LEARN(s,a) = true and $l(s,a) = m$. An attempted update that is not successful is an* **unsuccessful update**.

To prove the main theorem we need some additional results. The following lemmas are modified slightly from Strehl et al. (2006b). For convenience, define

$$\kappa := \frac{SA}{(1-\gamma)\varepsilon_1}.$$

Lemma 18 *The total number of updates during any execution of Delayed Q-learning is at most κ.*

Proof Consider a fixed state-action pair (s,a). Its associated action-value estimate $Q(s,a)$ is initialized to $U(s,a) \leq 1/(1-\gamma)$ before any updates occur. Each time $Q(s,a)$ is updated it decreases by at least ε_1. Since all rewards encountered are non-negative, the quantities involved in any update (see Equation 6) are non-negative. Thus, $Q(s,a)$ cannot fall below 0. It follows that $Q(s,a)$ cannot be updated more than $1/(\varepsilon(1-\gamma))$ times. Since there are SA state-action pairs, we have that there are at most $SA/(\varepsilon(1-\gamma))$ total updates. ■

Lemma 19 *The total number of attempted updates during any execution of Delayed Q-learning is at most $SA(1+\kappa)$.*

Proof Consider a fixed state-action pair (s,a). Once (s,a) is experienced for the m^{th} time, an attempted update will occur. Suppose that an attempted update of (s,a) occurs during timestep t. Afterwards, for another attempted update to occur during some later timestep t', it must be the case that a successful update of some state-action pair (not necessarily (s,a)) has occurred on or after timestep t and before timestep t'. From Lemma 18, there can be at most κ total successful updates. Therefore, there are at most $1+\kappa$ attempted updates of (s,a). Since there are SA state-action pairs, there can be at most $SA(1+\kappa)$ total attempted updates. ■

Definition 20 *During timestep t of the execution of Delayed Q-learning, we define K_t to be the set*

$$K_t := \left\{(s,a) \in \mathcal{S}\times\mathsf{A} \mid Q_t(s,a) - \left(R(s,a) + \gamma\sum_{s'} T(s'|s,a)V_t(s')\right) \leq 3\varepsilon_1\right\}.$$

The set K_t consists of the state-action pairs with low Bellman residual. The state-action pairs not in K_t are the ones whose action-value estimates are overly optimistic in the sense that they would decrease significantly if subjected to a Bellman backup (as in value iteration). Intuitively, if $(s,a) \notin K_t$, then it is very likely that (s,a) will be updated successfully by Delayed Q-learning if visited m times. This intuition is formalized by the following definition and lemma.

Definition 21 *Suppose we execute Delayed Q-learning in an MDP M. Define* **Event A2** *to be the event that for all timesteps t, if $(s,a) \notin K_{k_1}$ and an attempted update of (s,a) occurs during timestep t, then the update will be successful, where $k_1 < k_2 < \cdots < k_m = t$ are m last timesteps during which (s,a) is experienced consecutively by the agent.*

Lemma 22 *Suppose we execute Delayed Q-learning with parameter m satisfying*

$$m \geq \frac{(1+\gamma V_{\max})^2}{2{\varepsilon_1}^2} \ln\left(\frac{3SA}{\delta}\left(1+\frac{SA}{\varepsilon_1(1-\gamma)}\right)\right) \tag{8}$$

in an MDP M. The probability that Event A2 occurs is greater than or equal to $1-\delta/3$.

Proof Fix any timestep k_1 (and the complete history of the agent up to k_1) satisfying: the agent is in state s and about to take action a, where $(s,a) \notin K_{k_1}$ on timestep k_1, $LEARN(s,a) = true$, and $l(s,a) = 0$ at time k_1. In other words, if (s,a) is experienced $m-1$ more times after timestep k_1, then an attempted update will result. Let $Q = [(s[1],r[1]),\ldots,(s[m],r[m])] \in (S \times \mathbb{R})^m$ be any sequence of m next-state and immediate reward tuples. Due to the Markov assumption, whenever the agent is in state s and chooses action a, the resulting next-state and immediate reward are chosen independently of the history of the agent. Thus, the probability of the joint event

1. (s,a) is experienced $m-1$ more times, and
2. the resulting next-state and immediate reward sequence equals Q

is at most the probability that Q is obtained by m independent draws from the transition and reward distributions (for (s,a)). Therefore, it suffices to prove this lemma by showing that the probability that a random sequence Q could cause an unsuccessful update of (s,a) is at most $\delta/3$. We prove this statement next.

Suppose m rewards, $r[1],\ldots,r[m]$, and m next states, $s[1],\ldots,s[m]$, are drawn independently from the reward and transition distributions, respectively, for (s,a). By a straightforward application of the Hoeffding bound (with random variables $X_i := r[i] + \gamma V_{k_1}(s[i])$ so that $0 \leq X_i \leq (1+\gamma V_{\max})$), it can be shown that our choice of m guarantees that

$$\frac{1}{m}\sum_{i=1}^{m}(r[i]+\gamma V_{k_1}(s[i])) - \mathbf{E}[X_1] < \varepsilon_1$$

holds with probability at least $1-\delta/(3SA(1+\kappa))$. If it does hold and an attempted update is performed for (s,a) using these m samples, then the resulting update will succeed. To see the claim's validity, suppose that (s,a) is experienced at times $k_1 < k_2 < \cdots < k_m = t$ and at time k_i the agent is transitioned to state $s[i]$ and receives reward $r[i]$ (causing an attempted update at time t). Then, we have that

$$Q_t(s,a) - \left(\frac{1}{m}\sum_{i=1}^{m}(r[i]+\gamma V_{k_i}(s[i]))\right) \;>\; Q_t(s,a) - \mathbf{E}[X_1] - \varepsilon_1 \;>\; 2\varepsilon_1.$$

We have used the fact that $V_{k_i}(s') \leq V_{k_1}(s')$ for all s' and $i = 1, \ldots, m$. Therefore, with high probability, Equation 7 will be true and the attempted update of $Q(s,a)$ at time k_m will succeed.

Finally, we extend our argument, using the union bound, to all possible timesteps k_1 satisfying the condition above. The number of such timesteps is bounded by the same bound we showed for the number of attempted updates (that is, $SA(1+\kappa)$). ■

The next lemma states that, with high probability, Delayed Q-learning will maintain optimistic action values.

Lemma 23 *During execution of Delayed Q-learning, if m satisfies Equation 8, then $Q_t(s,a) \geq Q^*(s,a)$ holds for all timesteps t and state-action pairs (s,a), with probability at least $1-\delta/3$.*

Proof It can be shown, by a similar argument as in the proof of Lemma 22, that $(1/m)\sum_{i=1}^{m}(r_{k_i} + \gamma V^*(s_{k_i})) > Q^*(s,a) - \varepsilon_1$ holds, for all attempted updates, with probability at least $1-\delta/3$. Assuming this equation does hold, the proof is by induction on the timestep t. For the base case, note that $Q_1(s,a) = U(s,a) \geq Q^*(s,a)$ for all (s,a). Now, suppose the claim holds for all timesteps less than or equal to t. Thus, we have that $Q_t(s,a) \geq Q^*(s,a)$, and $V_t(s) \geq V^*(s)$ for all (s,a). Suppose s is the t^{th} state reached and a is the action taken at time t. If it does not result in an attempted update or it results in an unsuccessful update, then no action-value estimates change, and we are done. Otherwise, by Equation 6, we have that $Q_{t+1}(s,a) = (1/m)\sum_{i=1}^{m}(r_{k_i} + \gamma V_{k_i}(s_{k_i})) + \varepsilon_1 \geq (1/m)\sum_{i=1}^{m}(r_{k_i} + \gamma V^*(s_{k_i})) + \varepsilon_1 \geq Q^*(s,a)$, by the induction hypothesis and an application of the equation from above. ■

Lemma 24 *(Strehl et al., 2006b) If Event A2 occurs, then the following statement holds: If an unsuccessful update occurs at time t and $LEARN_{t+1}(s,a) = false$, then $(s,a) \in K_{t+1}$.*

Proof (By contradiction) Suppose an unsuccessful update occurs at timestep t, $LEARN_{t+1}(s,a) = false$, and $(s,a) \notin K_{t+1}$. Let $k_1 < k_2 < \cdots < k_m$ be the most recent m timesteps in which a is taken in state s. Clearly, $k_m = t$. Because of Event A2, we have $(s,a) \in K_{k_1}$. Since no update occurred on timestep t, we have that $K_t = K_{t+1}$. It follows from $K_t = K_{t+1}$ that $(s,a) \notin K_t$, implying that there must exist some timestep $t' > k_1$ in which a successful update occurs. Thus, by the rules of Section 2.2.2, $LEARN_{t+1}(s,a)$ remains $true$, which contradicts our assumption. ■

The following lemma bounds the number of timesteps t in which a state-action pair $(s,a) \notin K_t$ is experienced.

Lemma 25 *If Event A2 occurs and $Q_t(s,a) \geq Q^*(s,a)$ holds for all timesteps t and state-action pairs (s,a), then the number of changes to the Q-function is at most $\sum_{(s,a)\in S\times A} \frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$, and the number of timesteps t such that a state-action pair $(s_t,a_t) \notin K_t$ is at most $2m\sum_{(s,a)\in S\times A} \frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$.*

Proof We claim that $Q(s,a)$ cannot be changed more than $\frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$ times. First, note that $Q(s,a)$ is initialized to $U(s,a)$ and each successful update decreases its value by at least ε_1. Now, let $a^* = \text{argmax}_a Q^*(s,a)$. By assumption $Q(s,a^*) \geq Q^*(s,a^*) = V^*(s)$. Thus, we conclude that once

$Q(s,a)$ falls below $V^*(s)$, action a will never again be chosen in state s, since actions are chosen greedily with respect to $Q(\cdot,\cdot)$. Updates to (s,a) only occur after (s,a) has been experienced. Thus, at most $\frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$ changes to $Q(s,a)$ can occur, and the total number of changes to the Q-function is at most $\sum_{(s,a)\in\mathcal{S}\times\mathsf{A}} \frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$.

Suppose $(s,a)\notin K_t$ is experienced at time t and $LEARN_t(s,a) = \mathit{false}$ (implying the last attempted update was unsuccessful). By Lemma 24, we have that $(s,a)\in K_{t'+1}$ where t' was the time of the last attempted update of (s,a). Thus, some successful update has occurred since time $t'+1$. By the rules of Section 2.2.2, we have that $LEARN(s,a)$ will be set to *true* and by Event A2, the next attempted update will succeed.

Now, suppose that $(s,a)\notin K_t$ is experienced at time t and $LEARN_t(s,a) = \mathit{true}$. Within at most m more experiences of (s,a), an attempted update of (s,a) will occur. Suppose this attempted update takes place at time q and that the m most recent experiences of (s,a) happened at times $k_1 < k_2 < \cdots < k_m = q$. By Event A2, if $(s,a)\notin K_{k_1}$, the update will be successful. Otherwise, since $(s,a)\in K_{k_1}$, some successful update must have occurred between times k_1 and t (since $K_{k_1}\neq K_t$). Hence, even if the update is unsuccessful, $LEARN(s,a)$ will remain *true*, $(s,a)\notin K_{q+1}$ will hold, and the next attempted update of (s,a) will be successful.

In either case, if $(s,a)\notin K_t$, then within at most $2m$ more experiences of (s,a), a successful update of $Q(s,a)$ will occur. Thus, reaching a state-action pair not in K_t at time t will happen at most $2m\sum_{(s,a)\in\mathcal{S}\times\mathsf{A}} \frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$ times. ■

Using these Lemmas we can prove the main result.

Proof (of Theorem 16) We apply Theorem 10. Set m as in Lemma 22 and let $\varepsilon_1 = \varepsilon(1-\gamma)/3$. First, note that K_t is defined with respect to the agent's action-value estimates $Q(\cdot,\cdot)$ and other quantities that don't change during learning. Thus, we have that $K_t = K_{t+1}$ unless an update to some action-value estimate takes place. We now assume that Event A2 occurs, an assumption that holds with probability at least $1-\delta/3$, by Lemma 22. By Lemma 23, we have that Condition 1 of Theorem 10 holds, namely that $V_t(s)\geq V^*(s)-\varepsilon$ for all timesteps t. Next, we claim that Condition 2, $V_t(s)-V^{\pi_t}_{M_{K_t}}(s)\leq \frac{3\varepsilon_1}{1-\gamma}=\varepsilon$ also holds. For convenience let M' denote M_{K_t}. Recall that for all (s,a), either $Q_t(s,a)=Q^{\pi_t}_{M'}(s,a)$ when $(s,a)\notin K_t$, or $Q_t(s,a)-(R(s,a)+\gamma\sum_{s'}T(s'|s,a)V_t(s'))\leq 3\varepsilon_1$ when $(s,a)\in K_t$ (by definition of K_t). Note that $V^{\pi_t}_{M'}$ is the solution to the following set of Bellman equations:

$$\begin{aligned} V^{\pi_t}_{M'}(s) &= R(s,\pi_t(s))+\gamma\sum_{s'\in S} T(s'|s,\pi_t(s))V^{\pi_t}_{M'}(s') && \text{if } (s,\pi_t(s))\in K_t,\\ V^{\pi_t}_{M'}(s) &= Q_t(s,\pi_t(s)), && \text{if } (s,\pi_t(s))\notin K_t. \end{aligned}$$

The vector V_t is the solution to a similar set of equations except with some additional positive reward terms on the right-hand side for the case $(s,\pi_t(s))\in K_t$, each bounded by $3\varepsilon_1$, due to our definition of the set K_t. This fact implies that $V_t(s)-V^{\pi_t}_{M_{K_t}}(s)\leq\frac{3\varepsilon_1}{1-\gamma}$, as desired; see, e.g., Munos and Moore (2000) for a proof. Finally, for Condition 3 of Theorem 10, we note that by Lemma 25, $\zeta(\varepsilon,\delta) = O\left(2m\sum_{(s,a)\in\mathcal{S}\times\mathsf{A}}\frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}\right) = O\left(\frac{(1+\gamma V_{\max})^2\sum_{(s,a)\in\mathcal{S}\times\mathsf{A}}[U(s,a)-V^*(s)]_+}{\varepsilon^3(1-\gamma)^3}\ln\frac{SA}{\varepsilon\delta(1-\gamma)}\right)$, where $\zeta(\varepsilon,\delta)$ is the number of updates and escape events that occur during execution of Delayed Q-learning with inputs ε and δ (equivalently, with inputs ε_1 and m, which are derived from ε and δ). ■

We've proven upper bounds on the learning complexity of Delayed Q-learning and R-MAX. The analysis techniques are general and have proven useful in analyzing other related algorithms (Asmuth et al., 2008; Brunskill et al., 2008; Leffler et al., 2007; Strehl et al., 2007; Strehl and Littman, 2008a).

4. A New Lower Bound

The main result of this section (Theorem 26) is an improvement on published lower bounds for learning in MDPs. Existing results (Kakade, 2003) show a linear dependence on S and ε, but we find that a linearithmic on S and a quadratic dependence on ε are necessary for any reinforcement-learning algorithm $\mathcal{A}$ that satisfies the following assumptions:

- $\mathcal{A}_t$ is a deterministic policy at all timesteps t, and
- $\mathcal{A}_t$ and $\mathcal{A}_{t+1}$ can differ only in s_t; namely, the action-selection policy of the algorithm may change only in the most recently visited state.

Both assumptions are introduced to simplify our analysis. We anticipate the same lower bound to hold without these assumptions as they do not appear to restrict the power of an algorithm in the family of difficult-to-learn MDPs that we will describe soon. Also, while we choose to drop dependence on $1/(1-\gamma)$ in our new lower bound to facilitate a cleaner analysis, we believe it is possible to force a quadratic dependence by a more careful analysis. Finally, we note that the analysis bears some similarity to the lower bound analysis of Leffler et al. (2005) although their result is different and is for a different learning model.

Theorem 26 *For any reinforcement-learning algorithm $\mathcal{A}$ that satisfies the two assumptions above, there exists an MDP M such that the sample complexity of $\mathcal{A}$ in M is*

$$\Omega\left(\frac{SA}{\varepsilon^2}\ln\frac{S}{\delta}\right).$$

To prove this theorem, consider the family of MDPs depicted in Figure 1. The MDPs have $S = N+2$ states: $\mathcal{S} = \{1,2,\ldots,N,+,-\}$, and A actions. For convenience, denote by $[N]$ the set $\{1,2,\ldots,N\}$. Transitions from each state $i \in [N]$ are the same, so only the transitions from state 1 are depicted. One of the actions (the solid one) deterministically transports the agent to state $+$ with reward $0.5+\varepsilon$. Let a be any of the other $A-1$ actions (the dashed ones). From any state $i \in [N]$, taking a will transition to $+$ with reward 1 and probability p_{ia}, and to $-$ with reward 0 otherwise, where $p_{ia} \in \{0.5, 0.5+2\varepsilon\}$ are numbers very close to $0.5+\varepsilon$. Furthermore, for each i, there is at most one a such that $p_{ia} = 0.5+2\varepsilon$. Transitions from states $+$ and $-$ are identical: they simply reset the agent to one of the states in $[N]$ uniformly at random.

In fact, the MDP defined above can be viewed as N copies of a multi-armed bandit problem where the states $+$ and $-$ are dummy states for randomly resetting the agent to the next "real" state. Therefore, the optimal action in a state i is independent of the optimal action in any other state: it is the solid action if $p_{ia} = 0.5$ for all dashed actions a; otherwise, it is the dashed action a for which $p_{ia} = 0.5+2\varepsilon$. Intuitively, this MDP is hard to learn for exactly the same reason that a biased coin is hard to learn if the bias (that is, the probability of head after a coin toss) is close to 0.5.

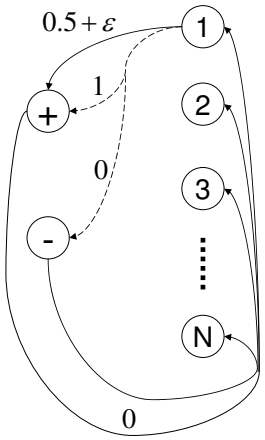


Figure 1: The difficult-to-learn MDPs for an improved sample complexity lower bound.

Lemma 27 *There exist constants $c_1, c_2 \in (0,1)$ such that during a whole run of the algorithm $\mathcal{A}$, for any state $i \in [N]$, the probability that $\mathcal{A}$ takes sub-optimal actions in i more than m_i times is at least $p(m_i)$, where*

$$p(m_i) := c_2 \exp\left(-\frac{m_i \varepsilon^2}{c_1 A}\right).$$

The following result is useful for proving Lemma 27.

Lemma 28 *(Mannor and Tsitsiklis, 2004, Theorem 1) Consider the K-armed bandit problem and let $\varepsilon, \delta \in (0,1)$. We call an algorithm $\mathcal{A}_B$ (ε,δ)-correct if it always terminates after a finite number T of trials and outputs an ε-optimal arm with probability at least $1-\delta$. Here, the sample complexity T is a random variable, and we let $\mathbf{E}$ be the expectation with respect to randomness in the bandit's rewards and $\mathcal{A}_B$ (if the algorithm is stochastic). Then there exist constants $c_1, c_2, \varepsilon_0, \delta_0 \in (0,1)$, such that for every $K \geq 2$, $\varepsilon \in (0, \varepsilon_0)$, and $\delta \in (0, \delta_0)$, and for every (ε,δ)-correct algorithm $\mathcal{A}_B$, there is a K-armed bandit problem such that*

$$\mathbf{E}[T] \geq \frac{c_1 K}{\varepsilon^2} \ln \frac{c_2}{\delta}.$$

Proof (of Lemma 27) If we treat decision making in each state as an A-arm bandit problem, finding the optimal action for that state becomes one of finding an ε-optimal arm (action) in the bandit problem. This bandit problem is the one used by Mannor and Tsitsiklis (2004) to establish the sample complexity lower bound in Lemma 28.[13]

By construction of the MDP in Figure 1, there is at most one optimal action in each state $i \in [N]$. Thus, if any RL algorithm $\mathcal{A}$ can guarantee, with probability at least $1-\delta_i$, that at most m_i sub-optimal actions are taken in state i during a whole run, then we can turn it into a bandit algorithm $\mathcal{A}_B$ with a sample complexity of $2m_i + 1$ in the following way: we simply run $\mathcal{A}$ for $2m_i + 1$ steps and the majority action must be ε-optimal with probability at least $1-\delta_i$. In other words, Lemma 28

13. The lower bound of Mannor and Tsitsiklis (2004) is for *expected* sample complexity. But, this result automatically applies to *worst-case* sample complexity, which is what we consider in the present paper.

for sample complexity in K-armed bandits results immediately in a lower bound for the total number of sub-optimal actions taken by $\mathcal{A}$, yielding

$$m_i \geq \frac{c_1 A}{\varepsilon^2} \ln \frac{c_2}{\delta_i}$$

for appropriately chosen constants c_1 and c_2. Reorganizing terms gives the desired result. ■

We will need two technical lemma to prove the lower bound. Their proofs are given after the proof of the main theorem.

Lemma 29 *Let c and Δ be constants in $(0,1)$. Under the constraints $\sum_i m_i \leq \zeta$ and $m_i > 0$ for all i, the function*

$$f(m_1, m_2, \ldots, m_N) = 1 - \prod_{i=1}^{N} (1 - c\Delta^{m_i})$$

is minimized when $m_1 = m_2 = \cdots = m_N = \frac{\zeta}{N}$. Therefore,

$$f(m_1, m_2, \ldots, m_N) \geq 1 - \left(1 - c\Delta^{\frac{\zeta}{N}}\right)^N.$$

Lemma 30 *If there exist some constants $c_1, c_2 > 0$ such that*

$$\delta \geq 1 - \left(1 - c_2 \exp\left(-\frac{\zeta\eta}{c_1\Psi}\right)\right)^{\Psi},$$

for some positive quantities η, ζ, Ψ, and δ, then

$$\zeta = \Omega\left(\frac{\Psi}{\eta} \ln \frac{\Psi}{\delta}\right).$$

Proof (of Theorem 26) Let $\zeta(\varepsilon, \delta)$ be an upper bound of the sample complexity of any PAC-MDP algorithm $\mathcal{A}$ with probability at least $1 - \delta$. Let sub-optimal actions be taken m_i times in state $i \in [N]$ during a whole run of $\mathcal{A}$. Consequently,

$$\delta \geq \Pr\left(\sum_{i=1}^{N} m_i > \zeta(\varepsilon, \delta)\right) = 1 - \Pr\left(\sum_{i=1}^{N} m_i \leq \zeta(\varepsilon, \delta)\right),$$

where the first step is because the actual sample complexity of $\mathcal{A}$ is at least $\sum_i m_i$.

We wish to find a lower bound for the last expression above by optimizing the values of m_i's subject to the constraint, $\sum_i m_i \leq \zeta(\varepsilon, \delta)$. Due to the statistical independence of what states $i \in [N]$ are visited by the algorithm,[14] we can factor the probability above to obtain

$$\delta \geq 1 - \max_{m_1, \ldots, m_N; \sum_i m_i \leq \zeta(\varepsilon, \delta)} \prod_{i=1}^{N} (1 - p(m_i)).$$

14. It does not help for the algorithm to base its policy in one state on samples collected in other states, due to the independence of states in this MDP. If an algorithm attempts to do so, an adversary can make use of this fact to assign p_{ia} to even *increase* the failure probability of the algorithm.

where Lemma 27 is applied.

We now use Lemma 29 to obtain a lower bound of the last expression above, which in turn lower-bounds δ. Applying this lemma with $c = c_2$ and $\Delta = \exp(-\frac{\varepsilon^2}{c_1 A})$ gives

$$\delta \geq 1 - \left(1 - c_2 \exp\left(-\frac{\zeta(\varepsilon,\delta)\varepsilon^2}{c_1 NA}\right)\right)^N. \tag{9}$$

The theorem then follows immediately from Lemma 30 using $\Psi = N$ and $\eta = \varepsilon^2/A$. ■

Proof (of Lemma 29) Since $f(m_1,\ldots,m_N) \in (0,1)$, finding the *minimum* of f is equivalent to finding the *maximum* of the following function:

$$g(m_1,m_2,\ldots,m_N) = \ln\left(1 - f(m_1,m_2,\ldots,m_N)\right) = \sum_{i=1}^{N} \ln\left(1 - c\Delta^{m_i}\right),$$

under the same constraints. Due to the concavity of $\ln(\cdot)$, we have

$$g(m_1,m_2,\ldots,m_N) \leq N \ln\left(\frac{1}{N}\sum_{i=1}^{N}\left(1 - c\Delta^{m_i}\right)\right) = N \ln\left(1 - \frac{c}{N}\sum_{i=1}^{N}\Delta^{m_i}\right).$$

Finally, we use the fact that the arithmetic mean is no less than the geometric mean to further simplify the upper bound of g:

$$g(m_1,m_2,\ldots,m_N) \leq N \ln\left(1 - c\Delta^{\frac{1}{N}\sum_{i=1}^{N} m_i}\right) \leq N \ln\left(1 - c\Delta^{\frac{\zeta}{N}}\right).$$

Equality holds in all inequalities above when $m_1 = m_2 = \cdots = m_N = \frac{\zeta}{N}$. ■

Proof (of Lemma 30) Reorganizing terms in Equation (9) gives

$$1 - c_2 \exp\left(-\frac{\zeta\eta}{c_1\Psi}\right) \geq (1-\delta)^{\frac{1}{\Psi}}.$$

The function $(1-\delta)^{1/\delta}$ is a decreasing function of δ for $0 < \delta < 1$, and $\lim_{\delta\to 0^+}(1-\delta)^{1/\delta} = 1/e$. Therefore, as long as δ is less than some constant $c_3 \in (0,1)$, we will have

$$(1-\delta)^{\frac{1}{\Psi}} = \left((1-\delta)^{\frac{1}{\delta}}\right)^{\frac{\delta}{\Psi}} \geq (c_4)^{\frac{\delta}{\Psi}} = \exp\left(-\frac{c_5\delta}{\Psi}\right),$$

where $c_4 = (1-c_3)^{1/c_3} \in \left(0,\frac{1}{e}\right)$ and $c_5 = \ln\frac{1}{c_4} \in (1,\infty)$ are two constants. It is important to note that c_3 (and thus c_4 and c_5) does not depend on η or Ψ. Now, apply the inequality $e^x \geq 1 + x$ for $x = -c_5\delta/\Psi$ to get $\exp(-c_5\delta/\Psi) \geq 1 - c_5\delta/\Psi$. The above chain of inequalities results in:

$$1 - c_2 \exp\left(-\frac{\zeta\eta}{c_1\Psi}\right) \geq 1 - \frac{c_5\delta}{\Psi}.$$

Solving this inequality for ζ gives the desired lower bound for ζ. ■

We have shown a new sample complexity lower bound that has a linearithmic dependence on S in the worst case. Thus, Delayed Q-learning is optimal in the sense of minimizing the dependence (of sample complexity of exploration) on the number of states.

5. Conclusion

We have presented and improved PAC-MDP upper and lower bounds reported in the literature. We studied two algorithms, R-MAX (which is model *based*) and Delayed Q-learning (which is model *free*) that are able to make use of non-trivial admissible heuristic functions. Comparing the relative strengths of model-based and model-free algorithms has been an important problem in the reinforcement-learning community (see, e.g., Atkeson and Gordon 1997 and Kearns and Singh 1999). Our analysis indicates that both can learn efficiently in finite MDPs in the PAC-MDP framework. The bounds suggest that a model-free method can be less sensitive on the size of the state space (linearithmic vs. quadratic dependence in the bound, matching the lower bound) whereas a model-based method can be less sensitive to the effective horizon, $1/(1-\gamma)$. Future work should focus on tightening bounds further and expanding analyses to state spaces in which generalization is necessary.

Acknowledgments

The authors appreciate supports from the National Science Foundation (IIS-0325281) and Rutgers University (Bevier fellowship). We thank John Langford, Eric Wiewiora, Sham Kakade, and Csaba Szepesvári for helpful discussions, especially on the analysis of Delayed Q-learning. We also thank the anonymous reviewers for their insightful comments that have significantly improved the quality of the paper.

References

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

John Asmuth, Michael L. Littman, and Robert Zinkov. Potential-based shaping in model-based reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 604–609. AAAI Press, 2008.

John Asmuth, Lihong Li, Michael L. Littman, Ali Nouri, and David Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

Christopher G. Atkeson and Geoffrey J. Gordon, editors. *Proceedings of the ICML-97 Workshop on Modelling in Reinforcement Learning*, 1997. URL `http://www.cs.cmu.edu/~ggordon/ml97ws/`.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. In *Advances in Neural Information Processing Systems 21*, pages 89–96, 2009.

Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

Ronen I. Brafman and Moshe Tennenholtz. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

Emma Brunskill, Bethany R. Leffler, Lihong Li, Michael L. Littman, and Nicholas Roy. CORL: A continuous-state offset-dynamics reinforcement learner. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 53–61, 2008. An extended version appears in the Journal of Machine Learning Research, volume 10, pages 1955–1988, 2009.

Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

Carlos Diuk, Lihong Li, and Bethany R. Leffler. The adaptive k-meteorologists problem and its application to structure discovery and feature selection in reinforcement learning. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning*, pages 249–256, 2009.

Michael O. Duff and Andrew G. Barto. Local bandit approximation for optimal learning problems. In *Advances in Neural Information Processing Systems*, volume 9, pages 1019–1025. The MIT Press, 1997.

Eyal Even-Dar and Yishay Mansour. Learning rates for Q-learning. *Journal of Machine Learning Research*, 5:1–25, 2003.

Claude-Nicolas Fiechter. Efficient reinforcement learning. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 88–97. Association of Computing Machinery, 1994.

John C. Gittins. *Multi-armed Bandit Allocation Indices*. Wiley Interscience Series in Systems and Optimization. John Wiley & Sons Inc, Chichester, NY, 1989.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Sham M. Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

Michael J. Kearns and Daphne Koller. Efficient reinforcement learning in factored MDPs. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 740–747, 1999.

Michael J. Kearns and Satinder P. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In *Advances in Neural Information Processing Systems 11*, pages 996–1002. The MIT Press, 1999.

Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2–3):209–232, 2002.

Sven Koenig and Reid G. Simmons. The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning*, 22(1–3):227–250, 1996.

Bethany R. Leffler, Michael L. Littman, Alexander L. Strehl, and Thomas J. Walsh. Efficient exploration with latent structure. In *Robotics: Science and Systems*, pages 81–88, 2005.

Bethany R. Leffler, Michael L. Littman, and Timothy Edmunds. Efficient reinforcement learning with relocatable action models. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, pages 572–577, 2007.

Lihong Li. *A Unifying Framework for Computational Reinforcement Learning Theory*. PhD thesis, Rutgers University, New Brunswick, NJ, 2009.

Lihong Li, Michael L. Littman, and Christopher R. Mansley. Online exploration in least-squares policy iteration. In *Proceedings of the Eighteenth International Conference on Agents and Multiagent Systems*, pages 733–739, 2009.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithms. *Machine Learning*, 2(4):285–318, 1988.

Shie Mannor and John N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5:623–648, 2004.

Colin McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics*, volume 141 of *London Mathematical Society Lecture Notes*, pages 148–188. Cambridge University Press, 1989.

Rémi Munos and Andrew W. Moore. Rates of convergence for variable resolution schemes in optimal control. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 647–654, 2000.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287, 1999.

Ali Nouri and Michael L. Littman. Multi-resolution exploration in continuous spaces. In *Advances in Neural Information Processing Systems 21*, pages 1209–1216, 2009.

Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 697–704, 2006.

Martin L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1994. ISBN 0-13-103805-2.

Satinder P. Singh and Richard C. Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.

Alexander L. Strehl and Michael L. Littman. A theoretical analysis of model-based interval estimation. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 857–864, 2005.

Alexander L. Strehl and Michael L. Littman. Online linear regression and its application to model-based reinforcement learning. In *Advances in Neural Information Processing Systems 20*, pages 1417–1424, 2008a.

Alexander L. Strehl and Michael L. Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008b.

Alexander L. Strehl, Lihong Li, and Michael L. Littman. Incremental model-based learners with formal learning-time guarantees. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 485–493, 2006a.

Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. PAC model-free reinforcement learning. In *Proceedings of the Twenty-Third International Conference on Machine learning*, pages 881–888, 2006b.

Alexander L. Strehl, Carlos Diuk, and Michael L. Littman. Efficient structure learning in factored-state MDPs. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 645–650, 2007.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Csaba Szepesvári. The asymptotic convergence-rate of Q-learning. In *Advances in Neural Information Processing Systems 10*, pages 1064–1070, 1998.

L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

Thomas J. Walsh, István Szita, Carlos Diuk, and Michael L. Littman. Exploring compact reinforcement-learning representations with linear regression. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J. Weinberger. Inequalities for the L1 deviation of the empirical distribution. Technical Report HPL-2003-97R1, Hewlett-Packard Labs, 2003.

Journal of Machine Learning Research 10 (2009) 2445-2471 Submitted 6/09; Published 11/09

Prediction With Expert Advice For The Brier Game

Vladimir Vovk VOVK@CS.RHUL.AC.UK
Fedor Zhdanov FEDOR@CS.RHUL.AC.UK
Computer Learning Research Centre
Department of Computer Science
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England

Editor: Yoav Freund

Abstract

We show that the Brier game of prediction is mixable and find the optimal learning rate and substitution function for it. The resulting prediction algorithm is applied to predict results of football and tennis matches, with well-known bookmakers playing the role of experts. The theoretical performance guarantee is not excessively loose on the football data set and is rather tight on the tennis data set.

Keywords: Brier game, classification, on-line prediction, strong aggregating algorithm, weighted average algorithm

1. Introduction

The paradigm of prediction with expert advice was introduced in the late 1980s (see, e.g., DeSantis et al., 1988, Littlestone and Warmuth, 1994, Cesa-Bianchi et al., 1997) and has been applied to various loss functions; see Cesa-Bianchi and Lugosi (2006) for a recent book-length review. An especially important class of loss functions is that of "mixable" ones, for which the learner's loss can be made as small as the best expert's loss plus a constant (depending on the number of experts). It is known (Haussler et al., 1998; Vovk, 1998) that the optimal additive constant is attained by the "strong aggregating algorithm" proposed in Vovk (1990) (we use the adjective "strong" to distinguish it from the "weak aggregating algorithm" of Kalnishkan and Vyugin, 2008).

There are several important loss functions that have been shown to be mixable and for which the optimal additive constant has been found. The prime examples in the case of binary observations are the log loss function and the square loss function. The log loss function, whose mixability is obvious, has been explored extensively, along with its important generalizations, the Kullback-Leibler divergence and Cover's loss function (see, e.g., the review by Vovk, 2001, Section 2.5).

In this paper we concentrate on the square loss function. In the binary case, its mixability was demonstrated in Vovk (1990). There are two natural directions in which this result could be generalized:

Regression: observations are real numbers (square-loss regression is a standard problem in statistics).

Classification: observations take values in a finite set (this leads to the "Brier game", to be defined shortly, a standard way of measuring the quality of predictions in meteorology and other applied fields: see, e.g., Dawid, 1986).

The mixability of the square loss function in the case of observations belonging to a bounded interval of real numbers was demonstrated in Haussler et al. (1998); Haussler et al.'s algorithm was simplified in Vovk (2001). Surprisingly, the case of square-loss non-binary classification has never been analysed in the framework of prediction with expert advice. The purpose of this paper is to fill this gap. Its short conference version (Vovk and Zhdanov, 2008a) appeared in the ICML 2008 proceedings.

2. Prediction Algorithm and Loss Bound

A game of prediction consists of three components: the observation space Ω, the decision space Γ, and the loss function $\lambda : \Omega \times \Gamma \to \mathbb{R}$. In this paper we are interested in the following *Brier game* (Brier, 1950): Ω is a finite and non-empty set, $\Gamma := \mathcal{P}(\Omega)$ is the set of all probability measures on Ω, and

$$\lambda(\omega, \gamma) = \sum_{o \in \Omega} (\gamma\{o\} - \delta_\omega\{o\})^2 ,$$

where $\delta_\omega \in \mathcal{P}(\Omega)$ is the probability measure concentrated at ω: $\delta_\omega\{\omega\} = 1$ and $\delta_\omega\{o\} = 0$ for $o \neq \omega$. (For example, if $\Omega = \{1,2,3\}$, $\omega = 1$, $\gamma\{1\} = 1/2$, $\gamma\{2\} = 1/4$, and $\gamma\{3\} = 1/4$, $\lambda(\omega,\gamma) = (1/2-1)^2 + (1/4-0)^2 + (1/4-0)^2 = 3/8$.)

The game of prediction is being played repeatedly by a learner having access to decisions made by a pool of experts, which leads to the following prediction protocol:

Protocol 1 Prediction with expert advice

$L_0 := 0$.
$L_0^k := 0, k = 1, \dots, K$.
for $N = 1, 2, \dots$ **do**
  Expert k announces $\gamma_N^k \in \Gamma, k = 1, \dots, K$.
  Learner announces $\gamma_N \in \Gamma$.
  Reality announces $\omega_N \in \Omega$.
  $L_N := L_{N-1} + \lambda(\omega_N, \gamma_N)$.
  $L_N^k := L_{N-1}^k + \lambda(\omega_N, \gamma_N^k), k = 1, \dots, K$.
end for

At each step of Protocol 1 Learner is given K experts' advice and is required to come up with his own decision; L_N is his cumulative loss over the first N steps, and L_N^k is the kth expert's cumulative loss over the first N steps. In the case of the Brier game, the decisions are probability forecasts for the next observation.

An optimal (in the sense of Theorem 1 below) strategy for Learner in prediction with expert advice for the Brier game is given by the strong aggregating algorithm (see Algorithm 1). For each expert k, the algorithm maintains its weight w^k, constantly slashing the weights of less successful experts. Its description uses the notation $t^+ := \max(t, 0)$.

The algorithm will be derived in Section 5. The following result (to be proved in Section 4) gives a performance guarantee for it that cannot be improved by any other prediction algorithm.

Algorithm 1 Strong aggregating algorithm for the Brier game

$w_0^k := 1, k = 1, \dots, K$.
for $N = 1, 2, \dots$ **do**
  Read the Experts' predictions $\gamma_N^k, k = 1, \dots, K$.
  Set $G_N(\omega) := -\ln \sum_{k=1}^K w_{N-1}^k e^{-\lambda(\omega, \gamma_N^k)}$, $\omega \in \Omega$.
  Solve $\sum_{\omega \in \Omega} (s - G_N(\omega))^+ = 2$ in $s \in \mathbb{R}$.
  Set $\gamma_N\{\omega\} := (s - G_N(\omega))^+ / 2$, $\omega \in \Omega$.
  Output prediction $\gamma_N \in \mathcal{P}(\Omega)$.
  Read observation ω_N.
  $w_N^k := w_{N-1}^k e^{-\lambda(\omega_N, \gamma_N^k)}$.
end for

Theorem 1 *Using Algorithm 1 as Learner's strategy in Protocol 1 for the Brier game guarantees that*

$$L_N \le \min_{k=1,\dots,K} L_N^k + \ln K \tag{1}$$

for all $N = 1, 2, \dots$. *If* $A < \ln K$, *Learner does not have a strategy guaranteeing*

$$L_N \le \min_{k=1,\dots,K} L_N^k + A \tag{2}$$

for all $N = 1, 2, \dots$.

3. Experimental Results

In our first empirical study of Algorithm 1 we use historical data about 8999 matches in various English football league competitions, namely: the Premier League (the pinnacle of the English football system), the Football League Championship, Football League One, Football League Two, the Football Conference. Our data, provided by Football-Data, cover four seasons, 2005/2006, 2006/2007, 2007/2008, and 2008/2009. The matches are sorted first by date, then by league, and then by the name of the home team. In the terminology of our prediction protocol, the outcome of each match is the observation, taking one of three possible values, "home win", "draw", or "away win"; we will encode the possible values as 1, 2, and 3.

For each match we have forecasts made by a range of bookmakers. We chose eight bookmakers for which we have enough data over a long period of time, namely Bet365, Bet&Win, Gamebookers, Interwetten, Ladbrokes, Sportingbet, Stan James, and VC Bet. (And the seasons mentioned above were chosen because the forecasts of these bookmakers are available for them.)

A probability forecast for the next observation is essentially a vector (p_1, p_2, p_3) consisting of positive numbers summing to 1. The bookmakers do not announce these numbers directly; instead, they quote three betting odds, a_1, a_2, and a_3. Each number $a_i > 1$ is the total amount which the bookmaker undertakes to pay out to a client betting on outcome i per unit stake in the event that i happens (if the bookmaker wishes to return the stake to the bettor, it should be included in a_i; i.e., the odds are announced according to the "continental" rather than "traditional" system). The inverse value $1/a_i$, $i \in \{1, 2, 3\}$, can be interpreted as the bookmaker's quoted probability for the observation i. The bookmaker's quoted probabilities are usually slightly (because of the competition with other bookmakers) in his favour: the sum $1/a_1 + 1/a_2 + 1/a_3$ exceeds 1 by the amount called

the *overround* (at most 0.15 in the vast majority of cases). We use Victor Khutsishvili's (2009) formula

$$p_i := a_i^{-\gamma}, \quad i = 1,2,3, \tag{3}$$

for computing the bookmaker's probability forecasts, where $\gamma > 0$ is chosen such that $a_1^{-\gamma} + a_2^{-\gamma} + a_3^{-\gamma} = 1$. Such a value of γ exists and is unique since the function $a_1^{-\gamma} + a_2^{-\gamma} + a_3^{-\gamma}$ continuously and strictly decreases from 3 to 0 as γ changes from 0 to ∞. In practice, we usually have $\gamma > 1$ as $a_1^{-1} + a_2^{-1} + a_3^{-1} > 1$ (i.e., the overround is positive). The method of bisection was more than sufficient for us to solve $a_1^{-\gamma} + a_2^{-\gamma} + a_3^{-\gamma} = 1$ with satisfactory accuracy. Khutsishvili's argument for (3) is outlined in Appendix B.

Typical values of γ in (3) are close to 1, and the difference $\gamma - 1$ reflects the bookmaker's target profit margin. In this respect $\gamma - 1$ is similar to the overround; indeed, the approximate value of the overround is $(\gamma - 1)\sum_{i=1}^{3} a_i^{-1} \ln a_i$ assuming that the overround is small and none of a_i is too close to 0. The coefficient of proportionality $\sum_{i=1}^{3} a_i^{-1} \ln a_i$ can be interpreted as the entropy of the quoted betting odds.

The results of applying Algorithm 1 to the football data, with 8 experts and 3 possible observations, are shown in Figure 1. Let L_N^k be the cumulative loss of Expert k, $k = 1, \dots, 8$, over the first N matches and L_N be the corresponding number for Algorithm 1 (i.e., we essentially continue to use the notation of Theorem 1). The dashed line corresponding to Expert k shows the excess loss $N \mapsto L_N^k - L_N$ of Expert k over Algorithm 1. The excess loss can be negative, but from the first part of Theorem 1 (Equation (1)) we know that it cannot be less than $-\ln 8$; this lower bound is also shown in Figure 1. Finally, the thick line (the positive part of the x axis) is drawn for comparison: this is the excess loss of Algorithm 1 over itself. We can see that at each moment in time the algorithm's cumulative loss is fairly close to the cumulative loss of the best expert (at that time; the best expert keeps changing over time).

Figure 2 shows the distribution of the bookmakers' overrounds. We can see that in most cases overrounds are between 0.05 and 0.15, but there are also occasional extreme values, near zero or in excess of 0.3.

Figure 3 shows the results of another empirical study, involving data about a large number of tennis tournaments in 2004, 2005, 2006, and 2007, with the total number of matches 10,087. The tournaments include, for example, Australian Open, French Open, US Open, and Wimbledon; the data is provided by Tennis-Data. The matches are sorted by date, then by tournament, and then by the winner's name. The data contain information about the winner of each match and the betting odds of 4 bookmakers for his/her win and for the opponent's win. Therefore, now there are two possible observations (player 1's win and player 2's win). There are four bookmakers: Bet365, Centrebet, Expekt, and Pinnacle Sports. The results in Figure 3 are presented in the same way as in Figure 1.

Typical values of the overround are below 0.1, as shown in Figure 4 (analogous to Figure 2).

In both Figure 1 and Figure 3 the cumulative loss of Algorithm 1 is close to the cumulative loss of the best expert. The theoretical bound is not hopelessly loose for the football data and is rather tight for the tennis data. The pictures look almost the same when Algorithm 1 is applied in the more realistic manner where the experts' weights w^k are not updated over the matches that are played simultaneously.

Our second empirical study (Figure 3) is about binary prediction, and so the algorithm of Vovk (1990) could have also been used (and would have given similar results). We included it since we are not aware of any empirical studies even for the binary case.

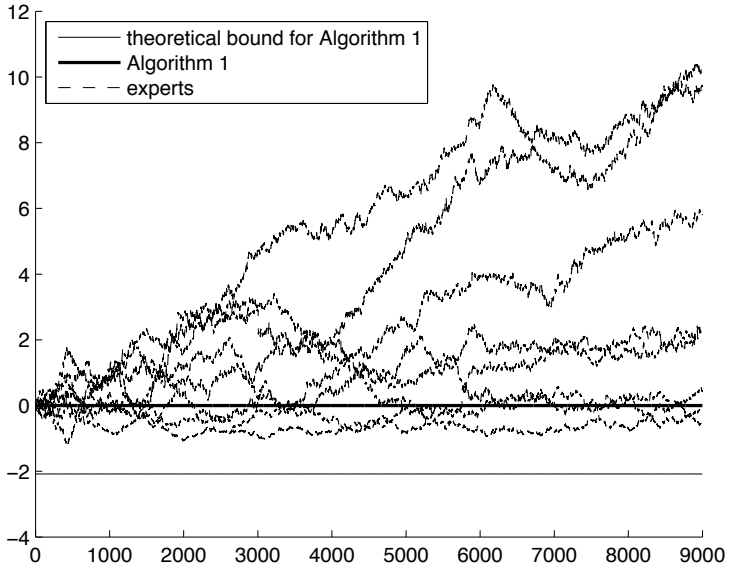


Figure 1: The difference between the cumulative loss of each of the 8 bookmakers (experts) and of Algorithm 1 on the football data. The theoretical lower bound $-\ln 8$ from Theorem 1 is also shown.

For comparison with several other popular prediction algorithms, see Appendix C. The data used for producing all the figures and tables in this section and in Appendix C can be downloaded from `http://vovk.net/ICML2008`.

4. Proof of Theorem 1

This proof will use some basic notions of elementary differential geometry, especially those connected with the Gauss-Kronecker curvature of surfaces. (The use of curvature in this kind of results is standard: see, e.g., Vovk, 1990, and Haussler et al., 1998.) All definitions that we will need can be found in, for example, Thorpe (1979).

A vector $f \in \mathbb{R}^\Omega$ (understood to be a function $f : \Omega \to \mathbb{R}$) is a *superprediction* if there is $\gamma \in \Gamma$ such that, for all $\omega \in \Omega$, $\lambda(\omega,\gamma) \le f(\omega)$; the set Σ of all superpredictions is the *superprediction set*. For each *learning rate* $\eta > 0$, let $\Phi_\eta : \mathbb{R}^\Omega \to (0,\infty)^\Omega$ be the homeomorphism defined by

$$\Phi_\eta(f) : \omega \in \Omega \mapsto e^{-\eta f(\omega)}, \quad f \in \mathbb{R}^\Omega. \tag{4}$$

The image $\Phi_\eta(\Sigma)$ of the superprediction set will be called the η-*exponential superprediction set*. It is known that

$$L_N \le \min_{k=1,\dots,K} L_N^k + \frac{\ln K}{\eta}, \quad N = 1,2,\dots,$$

can be guaranteed if and only if the η-exponential superprediction set is convex (part "if" for all K and part "only if" for $K \to \infty$ are proved in Vovk, 1998; part "only if" for all K is proved by Chris Watkins, and the details can be found in Appendix A). Comparing this with (1) and (2) we can see that we are required to prove that

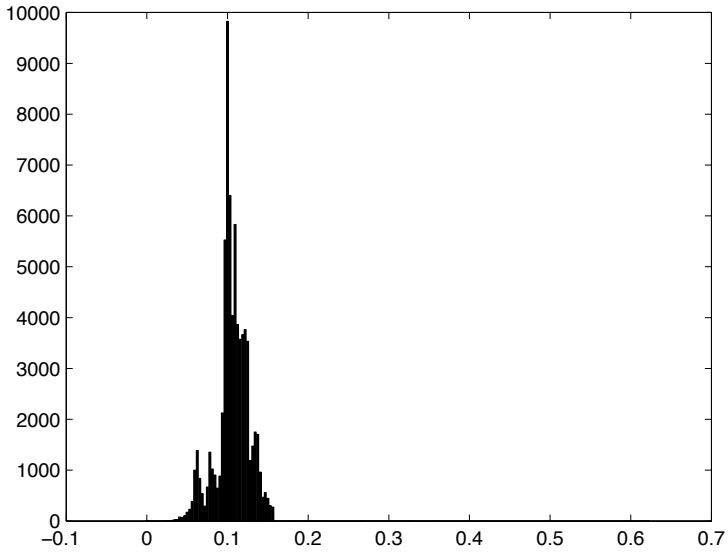


Figure 2: The overround distribution histogram for the football data, with 200 bins of equal size between the minimum and maximum values of the overround.

- $\Phi_\eta(\Sigma)$ is convex when $\eta \le 1$;
- $\Phi_\eta(\Sigma)$ is not convex when $\eta > 1$.

Define the η-*exponential superprediction surface* to be the part of the boundary of the η-exponential superprediction set $\Phi_\eta(\Sigma)$ lying inside $(0,\infty)^\Omega$. The idea of the proof is to check that, for all $\eta < 1$, the Gauss-Kronecker curvature of this surface is nowhere vanishing. Even when this is done, however, there is still uncertainty as to in which direction the surface is bulging (towards the origin or away from it). The standard argument (as in Thorpe, 1979, Chapter 12, Theorem 6) based on the continuity of the smallest principal curvature shows that the η-exponential superprediction set is bulging away from the origin for small enough η: indeed, since it is true at some point, it is true everywhere on the surface. By the continuity in η this is also true for all $\eta < 1$. Now, since the η-exponential superprediction set is convex for all $\eta < 1$, it is also convex for $\eta = 1$.

Let us now check that the Gauss-Kronecker curvature of the η-exponential superprediction surface is always positive when $\eta < 1$ and is sometimes negative when $\eta > 1$ (the rest of the proof, an elaboration of the above argument, will be easy). Set $n := |\Omega|$; without loss of generality we assume $\Omega = \{1,\dots,n\}$.

A convenient parametric representation of the η-exponential superprediction surface is

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} e^{-\eta((u_1-1)^2+u_2^2+\cdots+u_n^2)} \\ e^{-\eta(u_1^2+(u_2-1)^2+\cdots+u_n^2)} \\ \vdots \\ e^{-\eta(u_1^2+\cdots+(u_{n-1}-1)^2+u_n^2)} \\ e^{-\eta(u_1^2+\cdots+u_{n-1}^2+(u_n-1)^2)} \end{pmatrix}, \tag{5}$$

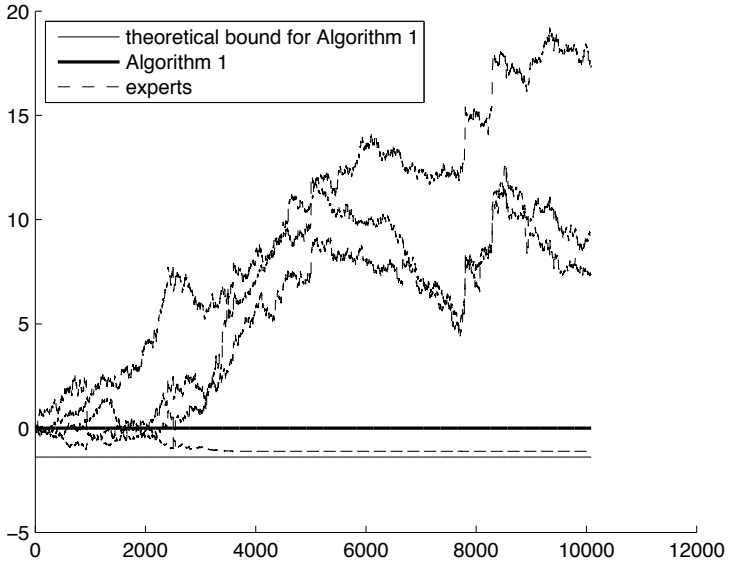


Figure 3: The difference between the cumulative loss of each of the 4 bookmakers and of Algorithm 1 on the tennis data. Now the theoretical bound is $-\ln 4$.

where $u_1,\dots,u_{n-1}$ are the coordinates on the surface, $u_1,\dots,u_{n-1}\in(0,1)$ subject to $u_1+\cdots u_{n-1}<1$, and u_n is a shorthand for $1-u_1-\cdots-u_{n-1}$. The derivative of (5) in u_1 is

$$\frac{\partial}{\partial u_1}\begin{pmatrix}x_1\\x_2\\\vdots\\x_{n-1}\\x_n\end{pmatrix}=2\eta\begin{pmatrix}(u_n-u_1+1)e^{-\eta((u_1-1)^2+u_2^2+\cdots+u_{n-1}^2+u_n^2)}\\(u_n-u_1)e^{-\eta(u_1^2+(u_2-1)^2+\cdots+u_{n-1}^2+u_n^2)}\\\vdots\\(u_n-u_1)e^{-\eta(u_1^2+u_2^2+\cdots+(u_{n-1}-1)^2+u_n^2)}\\(u_n-u_1-1)e^{-\eta(u_1^2+u_2^2+\cdots+u_{n-1}^2+(u_n-1)^2)}\end{pmatrix}\propto\begin{pmatrix}(u_n-u_1+1)e^{2\eta u_1}\\(u_n-u_1)e^{2\eta u_2}\\\vdots\\(u_n-u_1)e^{2\eta u_{n-1}}\\(u_n-u_1-1)e^{2\eta u_n}\end{pmatrix},$$

the derivative in u_2 is

$$\frac{\partial}{\partial u_2}\begin{pmatrix}x_1\\x_2\\\vdots\\x_{n-1}\\x_n\end{pmatrix}\propto\begin{pmatrix}(u_n-u_2)e^{2\eta u_1}\\(u_n-u_2+1)e^{2\eta u_2}\\\vdots\\(u_n-u_2)e^{2\eta u_{n-1}}\\(u_n-u_2-1)e^{2\eta u_n}\end{pmatrix},$$

and so on, up to

$$\frac{\partial}{\partial u_{n-1}}\begin{pmatrix}x_1\\x_2\\\vdots\\x_{n-1}\\x_n\end{pmatrix}\propto\begin{pmatrix}(u_n-u_{n-1})e^{2\eta u_1}\\(u_n-u_{n-1})e^{2\eta u_2}\\\vdots\\(u_n-u_{n-1}+1)e^{2\eta u_{n-1}}\\(u_n-u_{n-1}-1)e^{2\eta u_n}\end{pmatrix},$$

all coefficients of proportionality being equal and positive.

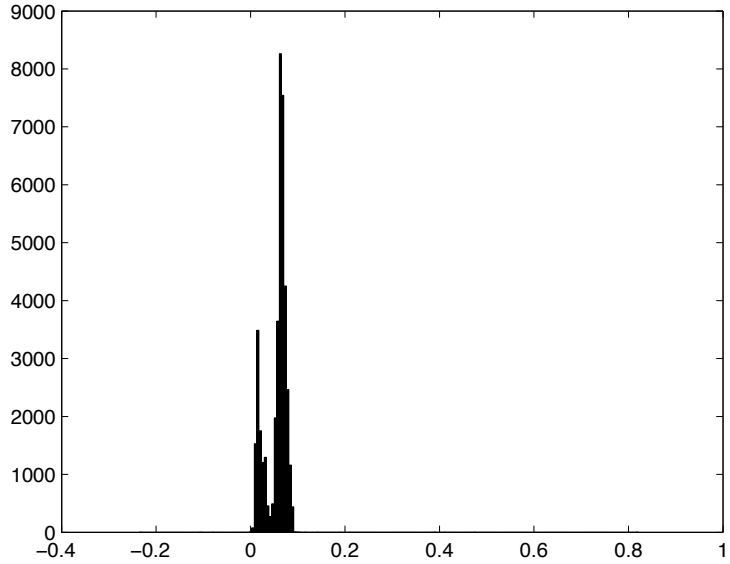


Figure 4: The overround distribution histogram for the tennis data.

A normal vector to the surface can be found as

$$Z := \begin{vmatrix} e_1 & \cdots & e_{n-1} & e_n \\ (u_n - u_1 + 1)e^{2\eta u_1} & \cdots & (u_n - u_1)e^{2\eta u_{n-1}} & (u_n - u_1 - 1)e^{2\eta u_n} \\ \vdots & \ddots & \vdots & \vdots \\ (u_n - u_{n-1})e^{2\eta u_1} & \cdots & (u_n - u_{n-1} + 1)e^{2\eta u_{n-1}} & (u_n - u_{n-1} - 1)e^{2\eta u_n} \end{vmatrix},$$

where e_i is the ith vector in the standard basis of $\mathbb{R}^n$ and $|\cdot|$ stands for the determinant (the matrix contains both scalars and vectors, but its determinant can still be computed using the standard rules). The coefficient in front of e_1 is the $(n-1)\times(n-1)$ determinant

$$\begin{vmatrix} (u_n - u_1)e^{2\eta u_2} & \cdots & (u_n - u_1)e^{2\eta u_{n-1}} & (u_n - u_1 - 1)e^{2\eta u_n} \\ (u_n - u_2 + 1)e^{2\eta u_2} & \cdots & (u_n - u_2)e^{2\eta u_{n-1}} & (u_n - u_2 - 1)e^{2\eta u_n} \\ \vdots & \ddots & \vdots & \vdots \\ (u_n - u_{n-1})e^{2\eta u_2} & \cdots & (u_n - u_{n-1} + 1)e^{2\eta u_{n-1}} & (u_n - u_{n-1} - 1)e^{2\eta u_n} \end{vmatrix}$$

$$\propto e^{-2\eta u_1} \begin{vmatrix} u_n - u_1 & \cdots & u_n - u_1 & u_n - u_1 - 1 \\ u_n - u_2 + 1 & \cdots & u_n - u_2 & u_n - u_2 - 1 \\ \vdots & \ddots & \vdots & \vdots \\ u_n - u_{n-1} & \cdots & u_n - u_{n-1} + 1 & u_n - u_{n-1} - 1 \end{vmatrix}$$

$$= e^{-2\eta u_1} \begin{vmatrix} 1 & 1 & \cdots & 1 & u_n - u_1 - 1 \\ 2 & 1 & \cdots & 1 & u_n - u_2 - 1 \\ 1 & 2 & \cdots & 1 & u_n - u_3 - 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 2 & u_n - u_{n-1} - 1 \end{vmatrix} = e^{-2\eta u_1} \begin{vmatrix} 1 & 1 & \cdots & 1 & u_n - u_1 - 1 \\ 1 & 0 & \cdots & 0 & u_1 - u_2 \\ 0 & 1 & \cdots & 0 & u_1 - u_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & u_1 - u_{n-1} \end{vmatrix}$$

$$
\begin{aligned}
&= e^{-2\eta u_1}\big((-1)^n(u_n-u_1-1)+(-1)^{n+1}(u_1-u_2)\\
&+(-1)^{n+1}(u_1-u_3)+\cdots+(-1)^{n+1}(u_1-u_{n-1})\big)\\
&= e^{-2\eta u_1}(-1)^n\big((u_2+u_3+\cdots+u_n)-(n-1)u_1-1\big)\\
&= -e^{-2\eta u_1}(-1)^n n u_1 \propto u_1 e^{-2\eta u_1} \qquad (6)
\end{aligned}
$$

(with a positive coefficient of proportionality, $e^{2\eta}$, in the first $\propto$; the third equality follows from the expansion of the determinant along the last column and then along the first row).

Similarly, the coefficient in front of e_i is proportional (with the same coefficient of proportionality) to $u_i e^{-2\eta u_i}$ for $i = 2,\dots,n-1$; indeed, the $(n-1)\times(n-1)$ determinant representing the coefficient in front of e_i can be reduced to the form analogous to (6) by moving the ith row to the top.

The coefficient in front of e_n is proportional to

$$
\begin{aligned}
&e^{-2\eta u_n}\begin{vmatrix}
u_n-u_1+1 & u_n-u_1 & \cdots & u_n-u_1 & u_n-u_1\\
u_n-u_2 & u_n-u_2+1 & \cdots & u_n-u_2 & u_n-u_2\\
\vdots & \vdots & \ddots & \vdots & \vdots\\
u_n-u_{n-2} & u_n-u_{n-2} & \cdots & u_n-u_{n-2}+1 & u_n-u_{n-2}\\
u_n-u_{n-1} & u_n-u_{n-1} & \cdots & u_n-u_{n-1} & u_n-u_{n-1}+1
\end{vmatrix}\\
&= e^{-2\eta u_n}\begin{vmatrix}
1 & 0 & \cdots & 0 & u_n-u_1\\
0 & 1 & \cdots & 0 & u_n-u_2\\
\vdots & \vdots & \ddots & \vdots & \vdots\\
0 & 0 & \cdots & 1 & u_n-u_{n-2}\\
-1 & -1 & \cdots & -1 & u_n-u_{n-1}+1
\end{vmatrix} = e^{-2\eta u_n}\begin{vmatrix}
1 & 0 & \cdots & 0 & u_n-u_1\\
0 & 1 & \cdots & 0 & u_n-u_2\\
\vdots & \vdots & \ddots & \vdots & \vdots\\
0 & 0 & \cdots & 1 & u_n-u_{n-2}\\
0 & 0 & \cdots & 0 & nu_n
\end{vmatrix}\\
&= nu_n e^{-2\eta u_n}
\end{aligned}
$$

(with the coefficient of proportionality $e^{2\eta}(-1)^{n-1}$).

The Gauss-Kronecker curvature at the point with coordinates $(u_1,\dots,u_{n-1})$ is proportional (with a positive coefficient of proportionality, possibly depending on the point) to

$$
\begin{vmatrix}
\frac{\partial Z^{\mathrm{T}}}{\partial u_1}\\
\vdots\\
\frac{\partial Z^{\mathrm{T}}}{\partial u_{n-1}}\\
Z^{\mathrm{T}}
\end{vmatrix} \qquad (7)
$$

(Thorpe, 1979, Chapter 12, Theorem 5, with $^{\mathrm{T}}$ standing for transposition).

A straightforward calculation allows us to rewrite determinant (7) (ignoring the positive coefficient $((-1)^{n-1}ne^{2\eta})^n$) as

$$
\begin{vmatrix}
(1-2\eta u_1)e^{-2\eta u_1} & 0 & \cdots & 0 & (2\eta u_n-1)e^{-2\eta u_n}\\
0 & (1-2\eta u_2)e^{-2\eta u_2} & \cdots & 0 & (2\eta u_n-1)e^{-2\eta u_n}\\
\vdots & \vdots & \ddots & \vdots & \vdots\\
0 & 0 & \cdots & (1-2\eta u_{n-1})e^{-2\eta u_{n-1}} & (2\eta u_n-1)e^{-2\eta u_n}\\
u_1 e^{-2\eta u_1} & u_2 e^{-2\eta u_2} & \cdots & u_{n-1}e^{-2\eta u_{n-1}} & u_n e^{-2\eta u_n}
\end{vmatrix}
$$

$$\propto \begin{vmatrix} 1-2\eta u_1 & 0 & \cdots & 0 & 2\eta u_n - 1 \\ 0 & 1-2\eta u_2 & \cdots & 0 & 2\eta u_n - 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1-2\eta u_{n-1} & 2\eta u_n - 1 \\ u_1 & u_2 & \cdots & u_{n-1} & u_n \end{vmatrix}$$

$$= u_1(1-2\eta u_2)(1-2\eta u_3)\cdots(1-2\eta u_n) + u_2(1-2\eta u_1)(1-2\eta u_3)\cdots(1-2\eta u_n) + \cdots + u_n(1-2\eta u_1)(1-2\eta u_2)\cdots(1-2\eta u_{n-1}) \quad (8)$$

(with a positive coefficient of proportionality; to avoid calculation of the parities of various permutations, the reader might prefer to prove the last equality by induction in n, expanding the last determinant along the first column). Our next goal is to show that the last expression in (8) is positive when $\eta < 1$ but can be negative when $\eta > 1$.

If $\eta > 1$, set $u_1 = u_2 := 1/2$ and $u_3 = \cdots = u_n := 0$. The last expression in (8) becomes negative. It will remain negative if u_1 and u_2 are sufficiently close to $1/2$ and $u_3, \ldots, u_n$ are sufficiently close to 0.

It remains to consider the case $\eta < 1$. Set $t_i := 1 - 2\eta u_i$, $i = 1, \ldots, n$; the constraints on the t_i are

$$\begin{gathered} -1 < 1 - 2\eta < t_i < 1, \quad i = 1, \ldots, n, \\ t_1 + \cdots + t_n = n - 2\eta > n - 2. \end{gathered} \quad (9)$$

Our goal is to prove

$$(1-t_1)t_2t_3\cdots t_n + \cdots + (1-t_n)t_1t_2\cdots t_{n-1} > 0,$$

that is,

$$t_2t_3\cdots t_n + \cdots + t_1t_2\cdots t_{n-1} > nt_1\cdots t_n. \quad (10)$$

This reduces to

$$\frac{1}{t_1} + \cdots + \frac{1}{t_n} > n \quad (11)$$

if $t_1 \cdots t_n > 0$, and to

$$\frac{1}{t_1} + \cdots + \frac{1}{t_n} < n \quad (12)$$

if $t_1 \cdots t_n < 0$. The remaining case is where some of the t_i are zero; for concreteness, let $t_n = 0$. By (9) we have $t_1 + \cdots + t_{n-1} > n-2$, and so all of $t_1, \ldots, t_{n-1}$ are positive; this shows that (10) is indeed true.

Let us prove (11). Since $t_1 \cdots t_n > 0$, all of $t_1, \ldots, t_n$ are positive (if two of them were negative, the sum $t_1 + \cdots + t_n$ would be less than $n-2$; cf. (9)). Therefore,

$$\frac{1}{t_1} + \cdots + \frac{1}{t_n} > \underbrace{1 + \cdots + 1}_{n \text{ times}} = n.$$

To establish (10) it remains to prove (12). Suppose, without loss of generality, that $t_1 > 0$, $t_2 > 0, \ldots, t_{n-1} > 0$, and $t_n < 0$. We will prove a slightly stronger statement allowing $t_1, \ldots, t_{n-2}$ to take value 1 and removing the lower bound on t_n. Since the function $t \in (0,1] \mapsto 1/t$ is convex, we can also assume, without loss of generality, $t_1 = \cdots = t_{n-2} = 1$. Then $t_{n-1} + t_n > 0$, and so

$$\frac{1}{t_{n-1}} + \frac{1}{t_n} < 0;$$

therefore,

$$\frac{1}{t_1}+\cdots+\frac{1}{t_{n-2}}+\frac{1}{t_{n-1}}+\frac{1}{t_n}<n-2<n.$$

Finally, let us check that the positivity of the Gauss-Kronecker curvature implies the convexity of the η-exponential superprediction set in the case $\eta\le 1$, and the lack of positivity of the Gauss-Kronecker curvature implies the lack of convexity of the η-exponential superprediction set in the case $\eta>1$. The η-exponential superprediction surface will be oriented by choosing the normal vector field directed towards the origin. This can be done since

$$\begin{pmatrix}x_1\\ \vdots\\ x_n\end{pmatrix}\propto\begin{pmatrix}e^{2\eta u_1}\\ \vdots\\ e^{2\eta u_n}\end{pmatrix},\quad Z\propto(-1)^{n-1}\begin{pmatrix}u_1e^{-2\eta u_1}\\ \vdots\\ u_ne^{-2\eta u_n}\end{pmatrix},\tag{13}$$

with both coefficients of proportionality positive (cf. (5) and the bottom row of the first determinant in (8)), and the sign of the scalar product of the two vectors on the right-hand sides in (13) does not depend on the point $(u_1,\dots,u_{n-1})$. Namely, we take $(-1)^nZ$ as the normal vector field directed towards the origin. The Gauss-Kronecker curvature will not change sign after the re-orientation: if n is even, the new orientation coincides with the old, and for odd n the Gauss-Kronecker curvature does not depend on the orientation.

In the case $\eta>1$, the Gauss-Kronecker curvature is negative at some point, and so the η-exponential superprediction set is not convex (Thorpe, 1979, Chapter 13, Theorem 1 and its proof).

It remains to consider the case $\eta\le 1$. Because of the continuity of the η-exponential superprediction surface in η we can and will assume, without loss of generality, that $\eta<1$.

Let us first check that the smallest principal curvature $k_1=k_1(u_1,\dots,u_{n-1},\eta)$ of the η-exponential superprediction surface is always positive (among the arguments of k_1 we list not only the coordinates $u_1,\dots,u_{n-1}$ of a point on the surface (5) but also the learning rate $\eta\in(0,1)$). At least at some $(u_1,\dots,u_{n-1},\eta)$ the value of $k_1(u_1,\dots,u_{n-1},\eta)$ is positive: take a sufficiently small η and the point on the surface (5) with coordinates $u_1=\dots=u_{n-1}=1/n$; a simple calculation shows that this point will be a point of local maximum for $x_1+\dots+x_n$. Therefore, for all $(u_1,\dots,u_{n-1},\eta)$ the value of $k_1(u_1,\dots,u_{n-1},\eta)$ is positive: if k_1 had different signs at two points in the set

$$\left\{(u_1,\dots,u_{n-1},\eta)\,|\,u_1\in(0,1),\dots,u_{n-1}\in(0,1),u_1+\dots+u_{n-1}<1,\eta\in(0,1)\right\},\tag{14}$$

we could connect these points by a continuous curve lying completely inside (14); at some point on the curve, k_1 would be zero, in contradiction to the positivity of the Gauss-Kronecker curvature $k_1\cdots k_{n-1}$.

Now it is easy to show that the η-exponential superprediction set is convex. Suppose there are two points A and B on the η-exponential superprediction surface such that the interval $[A,B]$ contains points outside the η-exponential superprediction set. The intersection of the plane OAB, where O is the origin, with the η-exponential superprediction surface is a planar curve; the curvature of this curve at some point between A and B will be negative (remember that the curve is oriented by directing the normal vector field towards the origin), contradicting the positivity of k_1 at that point.

5. Derivation of the Prediction Algorithm

To achieve the loss bound (1) in Theorem 1 Learner can use, as discussed earlier, the strong aggregating algorithm (see, e.g., Vovk, 2001, Section 2.1, (15)) with $\eta=1$. In this section we will find

a substitution function for the strong aggregating algorithm for the Brier game with $\eta \le 1$, which is the only component of the algorithm not described explicitly in Vovk (2001). Our substitution function will not require that its input, the generalized prediction, should be computed from the normalized distribution $(w^k)_{k=1}^K$ on the experts; this is a valuable feature for generalizations to an infinite number of experts (as demonstrated in, e.g., Vovk, 2001, Appendix A.1).

Suppose that we are given a generalized prediction $(l_1,\dots,l_n)^{\mathrm{T}}$ computed by the aggregating pseudo-algorithm from a normalized distribution on the experts. Since $(l_1,\dots,l_n)^{\mathrm{T}}$ is a superprediction (remember that we are assuming $\eta \le 1$), we are only required to find a permitted prediction

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix} = \begin{pmatrix} (u_1-1)^2+u_2^2+\dots+u_n^2 \\ u_1^2+(u_2-1)^2+\dots+u_n^2 \\ \vdots \\ u_1^2+u_2^2+\dots+(u_n-1)^2 \end{pmatrix} \tag{15}$$

(cf. (5)) satisfying

$$\lambda_1 \le l_1,\dots,\lambda_n \le l_n. \tag{16}$$

Now suppose we are given a generalized prediction $(L_1,\dots,L_n)^{\mathrm{T}}$ computed by the aggregating pseudo-algorithm from an unnormalized distribution on the experts; in other words, we are given

$$\begin{pmatrix} L_1 \\ \vdots \\ L_n \end{pmatrix} = \begin{pmatrix} l_1+c \\ \vdots \\ l_n+c \end{pmatrix}$$

for some $c \in \mathbb{R}$. To find (15) satisfying (16) we can first find the largest $t \in \mathbb{R}$ such that $(L_1 - t,\dots,L_n-t)^{\mathrm{T}}$ is still a superprediction and then find (15) satisfying

$$\lambda_1 \le L_1 - t,\dots,\lambda_n \le L_n - t. \tag{17}$$

Since $t \ge c$, it is clear that $(\lambda_1,\dots,\lambda_n)^{\mathrm{T}}$ will also satisfy the required (16).

Proposition 2 *Define $s \in \mathbb{R}$ by the requirement*

$$\sum_{i=1}^n (s-L_i)^+ = 2. \tag{18}$$

The unique solution to the optimization problem $t \to \max$ under the constraints (17) with $\lambda_1,\dots,\lambda_n$ as in (15) will be

$$u_i = \frac{(s-L_i)^+}{2}, \quad i=1,\dots,n, \tag{19}$$

$$t = s - 1 - u_1^2 - \dots - u_n^2. \tag{20}$$

There exists a unique s satisfying (18) since the left-hand side of (18) is a continuous, increasing (strictly increasing when positive) and unbounded above function of s. The substitution function is given by (19).

Proof of Proposition 2 Let us denote the u_i and t defined by (19) and (20) as $\overline{u}_i$ and $\overline{t}$, respectively. To see that they satisfy the constraints (17), notice that the ith constraint can be spelt out as

$$\overline{u}_1^2 + \cdots + \overline{u}_n^2 - 2\overline{u}_i + 1 \le L_i - \overline{t},$$

which immediately follows from (19) and (20). As a by-product, we can see that the inequality becomes an equality, that is,

$$\overline{t} = L_i - 1 + 2\overline{u}_i - \overline{u}_1^2 - \cdots - \overline{u}_n^2, \tag{21}$$

for all i with $\overline{u}_i > 0$.

We can rewrite (17) as

$$\begin{cases} t \le L_1 - 1 + 2u_1 - u_1^2 - \cdots - u_n^2, \\ \vdots \\ t \le L_n - 1 + 2u_n - u_1^2 - \cdots - u_n^2, \end{cases} \tag{22}$$

and our goal is to prove that these inequalities imply $t < \overline{t}$ (unless $u_1 = \overline{u}_1, \ldots, u_n = \overline{u}_n$). Choose $\overline{u}_i$ (necessarily $\overline{u}_i > 0$ unless $u_1 = \overline{u}_1, \ldots, u_n = \overline{u}_n$; in the latter case, however, we can, and will, also choose $\overline{u}_i > 0$) for which $\varepsilon_i := \overline{u}_i - u_i$ is maximal. Then every value of t satisfying (22) will also satisfy

$$\begin{aligned} t &\le L_i - 1 + 2u_i - \sum_{j=1}^n u_j^2 \\ &= L_i - 1 + 2\overline{u}_i - 2\varepsilon_i - \sum_{j=1}^n \overline{u}_j^2 + 2\sum_{j=1}^n \varepsilon_j \overline{u}_j - \sum_{j=1}^n \varepsilon_j^2 \\ &\le L_i - 1 + 2\overline{u}_i - \sum_{j=1}^n \overline{u}_j^2 - \sum_{j=1}^n \varepsilon_j^2 \le \overline{t}. \end{aligned} \tag{23}$$

The penultimate $\le$ in (23) follows from

$$-\varepsilon_i + \sum_{j=1}^n \varepsilon_j \overline{u}_j = \sum_{j=1}^n (\varepsilon_j - \varepsilon_i)\overline{u}_j \le 0.$$

The last $\le$ in (23) follows from (21) and becomes $<$ when not all u_j coincide with $\overline{u}_j$. ■

The detailed description of the resulting prediction algorithm was given as Algorithm 1 in Section 2. As discussed, that algorithm uses the generalized prediction $G_N(\omega)$ computed from unnormalized weights.

6. Conclusion

In this paper we only considered the simplest prediction problem for the Brier game: competing with a finite pool of experts. In the case of square-loss regression, it is possible to find efficient closed-form prediction algorithms competitive with linear functions (see, e.g., Cesa-Bianchi and Lugosi, 2006, Chapter 11). Such algorithms can often be "kernelized" to obtain prediction algorithms competitive with reproducing kernel Hilbert spaces of prediction rules. This would be an appealing research programme in the case of the Brier game as well.

Acknowledgments

Victor Khutsishvili has been extremely generous in sharing his ideas with us. Comments by Alexey Chernov, Yuri Kalnishkan, Alex Gammerman, Bob Vickers, and the anonymous referees for the conference and journal versions have helped us improve the presentation. The referees for the conference version also suggested comparing our results with the Weighted Average Algorithm and the Hedge algorithm. We are grateful to Football-Data and Tennis-Data for providing access to the data used in this paper. This work was supported in part by EPSRC (grant EP/F002998/1).

Appendix A. Watkins's Theorem

Watkins's theorem is stated in Vovk (1999, Theorem 8) not in sufficient generality: it presupposes that the loss function is mixable. The proof, however, shows that this assumption is irrelevant (it can be made part of the conclusion), and the goal of this appendix is to give a self-contained statement of a suitable version of the theorem. (The reader will notice that the generality of the new version is essential only for our discussion in Section 4, not for Theorem 1 itself.)

In this appendix we will use a slightly more general notion of a game of prediction $(\Omega, \Gamma, \lambda)$: namely, the loss function $\lambda : \Omega \times \Gamma \to \overline{\mathbb{R}}$ is now allowed to take values in the extended real line $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, \infty\}$ (although the value $-\infty$ will be later disallowed).

Partly following Vovk (1998), for each $K = 1, 2, \ldots$ and each $a > 0$ we consider the following perfect-information game $\mathcal{G}_K(a)$ (the "global game") between two players, Learner and Environment. Environment is a team of $K+1$ players called Expert 1 to Expert K and Reality, who play with Learner according to Protocol 1. Learner wins if, for all $N = 1, 2, \ldots$ and all $k \in \{1, \ldots, K\}$,

$$L_N \le L_N^k + a; \tag{24}$$

otherwise, Environment wins. It is possible that $L_N = \infty$ or $L_N^k = \infty$ in (24); the interpretation of inequalities involving infinities is natural.

For each K we will be interested in the set of those $a > 0$ for which Learner has a winning strategy in the game $\mathcal{G}_K(a)$ (we will denote this by $\mathrm{L} \smile \mathcal{G}_K(a)$). It is obvious that

$$\mathrm{L} \smile \mathcal{G}_K(a) \,\&\, a' > a \Longrightarrow \mathrm{L} \smile \mathcal{G}_K(a');$$

therefore, for each K there exists a unique *borderline value* a_K such that $\mathrm{L} \smile \mathcal{G}_K(a)$ holds when $a > a_K$ and fails when $a < a_K$. It is possible that $a_K = \infty$ (but remember that we are only interested in finite values of a).

These are our assumptions about the game of prediction (similar to those in Vovk, 1998):

- Γ is a compact topological space;
- for each $\omega \in \Omega$, the function $\gamma \in \Gamma \mapsto \lambda(\omega, \gamma)$ is continuous ($\overline{\mathbb{R}}$ is equipped with the standard topology);
- there exists $\gamma \in \Gamma$ such that, for all $\omega \in \Omega$, $\lambda(\omega, \gamma) < \infty$;
- the function λ is bounded below.

We say that the game of prediction $(\Omega, \Gamma, \lambda)$ is η-*mixable*, where $\eta > 0$, if

$$\forall \gamma_1 \in \Gamma, \gamma_2 \in \Gamma, \alpha \in [0,1]\ \exists \delta \in \Gamma\ \forall \omega \in \Omega\colon\ e^{-\eta\lambda(\omega,\delta)} \geq \alpha e^{-\eta\lambda(\omega,\gamma_1)} + (1-\alpha)e^{-\eta\lambda(\omega,\gamma_2)}. \tag{25}$$

In the case of finite Ω, this condition says that the image of the superprediction set under the mapping Φ_η (see (4)) is convex. The game of prediction is *mixable* if it is η-mixable for some $\eta > 0$.

It follows from Hardy et al. (1952, Theorem 92, applied to the means $\mathfrak{M}_\phi$ with $\phi(x) = e^{-\eta x}$) that if the prediction game is η-mixable it will remain η'-mixable for any positive $\eta' < \eta$. (For another proof, see the end of the proof of Lemma 9 in Vovk, 1998.) Let η^* be the supremum of the η for which the prediction game is η-mixable (with $\eta^* := 0$ when the game is not mixable). The compactness of Γ implies that the prediction game is η^*-mixable.

Theorem 3 (Chris Watkins) *For any $K \in \{2,3,\ldots\}$,*

$$a_K = \frac{\ln K}{\eta^*}.$$

In particular, $a_K < \infty$ if and only if the game is mixable.

The theorem does not say explicitly, but it is easy to check, that $\mathrm{L} \smile \mathcal{G}_K(a_K)$: this follows both from general considerations (cf. Lemma 3 in Vovk, 1998) and from the fact that the strong aggregating algorithm wins $\mathcal{G}_K(a_K) = \mathcal{G}_K(\ln K/\eta^*)$.

Proof of Theorem 3 The proof will use some notions and notation used in the statement and proof of Theorem 1 of Vovk (1998). Without loss of generality we can, and will, assume that the loss function satisfies $\lambda > 1$ (add a suitable constant to λ if needed). Therefore, Assumption 4 of Vovk (1998) (the only assumption in that paper not directly made here) is satisfied. In view of the fact that $\mathrm{L} \smile \mathcal{G}_K(\ln K/\eta^*)$, we only need to show that $\mathrm{L} \smile \mathcal{G}_K(a)$ does not hold for $a < \ln K/\eta^*$. Fix $a < \ln K/\eta^*$.

The separation curve consists of the points $(c(\beta), c(\beta)/\eta) \in [0,\infty)^2$, where $\beta := e^{-\eta}$ and η ranges over $[0,\infty]$ (see Vovk, 1998, Theorem 1). Since the two-fold convex mixture in (25) can be replaced by any finite convex mixture (apply two-fold mixtures repeatedly), setting $\eta := \eta^*$ shows that the point $(1, 1/\eta^*)$ is Northeast of (actually belongs to) the separation curve. On the other hand, the point $(1, a/\ln K)$ is Southwest and outside of the separation curve (use Lemmas 8–12 of Vovk, 1998). Therefore, E (i.e., Environment) has a winning strategy in the game $\mathcal{G}(1, a/\ln K)$. It is easy to see from the proof of Theorem 1 in Vovk (1998) that the definition of the game $\mathcal{G}$ can be modified, without changing the conclusion about $\mathcal{G}(1, a/\ln K)$, by replacing the line

E chooses $n \geq 1$ {size of the pool}

in the protocol on p. 153 of Vovk (1998) by

E chooses $n^* \geq 1$ {lower bound on the size of the pool}

L chooses $n \geq n^*$ {size of the pool}

(indeed, the proof in Section 6 of Vovk, 1998, only requires that there should be sufficiently many experts). Let n^* be the first move by Environment according to her winning strategy.

Now suppose $\mathrm{L} \smile \mathcal{G}_K(a)$. From the fact that there exists Learner's strategy $\mathcal{L}_1$ winning $\mathcal{G}_K(a)$ we can deduce: there exists Learner's strategy $\mathcal{L}_2$ winning $\mathcal{G}_{K^2}(2a)$ (we can split the K^2 experts into K groups of K, merge the experts' decisions in each group with $\mathcal{L}_1$, and finally merge the groups' decisions with $\mathcal{L}_1$); there exists Learner's strategy $\mathcal{L}_3$ winning $\mathcal{G}_{K^3}(3a)$ (we can split the K^3 experts

Loss resulting from (3)	Loss resulting from (26)	Difference
5585.69	5588.20	2.52
5585.94	5586.67	0.72
5586.60	5587.37	0.77
5588.47	5590.65	2.18
5588.61	5589.92	1.31
5591.97	5593.48	1.52
5596.01	5601.85	5.84
5596.56	5598.02	1.46

Table 1: The bookmakers' cumulative Brier losses over the football data set when their probability forecasts are computed using formula (3) and formula (26).

into K groups of K^2, merge the experts' decisions in each group with $\mathcal{L}_2$, and finally merge the groups' decisions with $\mathcal{L}_1$); and so on. When the number K^m of experts exceeds n^*, we obtain a contradiction: Learner can guarantee

$$L_N \le L_N^k + ma$$

for all N and all K^m experts k, and Environment can guarantee that

$$L_N > L_N^k + \frac{a}{\ln K}\ln(K^m) = L_N^k + ma$$

for some N and k. ■

Appendix B. Khutsishvili's Theory

In the conference version of this paper (Vovk and Zhdanov, 2008a) we used

$$p_i := \frac{1/a_i}{1/a_1 + 1/a_2 + 1/a_3}, \quad i = 1,2,3, \tag{26}$$

in place of (3). A natural way to compare formulas (3) and (26) is to compare the losses of the probability forecasts found from the bookmakers' betting odds using those formulas. Using Khutsishvili's formula (3) consistently leads to smaller losses as measured by the Brier loss function: see Tables 1 and 2. The improvement of each bookmaker's total loss over the football data set is in the range 0.72–5.84; over the tennis data set the difference is in the range 1.27–11.64. These differences are of the order of the differences in cumulative loss between different bookmakers, and so the improvement is significant.

The goal of this appendix is to present, in a rudimentary form, Khutsishvili's theory behind (3). The theory is based on a very idealized model of a bookmaker, who is assumed to compute the betting odds a for an event of probability p using a function f,

$$a := f(p).$$

Loss resulting from (3)	Loss resulting from (26)	Difference
3935.32	3944.02	8.69
3943.83	3945.10	1.27
3945.70	3957.33	11.64
3953.83	3957.75	3.92

Table 2: The bookmakers' cumulative Brier losses over the tennis data set when their probability forecasts are computed using formula (3) and formula (26).

Different bookmakers (and the same bookmaker at different times) can use different functions f. Therefore, different bookmakers may quote different odds because they may use different f and because they may assign different probabilities to the same event.

The following simple corollary of Darboux's theorem describes the set of possible functions f; its interpretation will be discussed straight after the proof.

Theorem 4 (Victor Khutsishvili) *Suppose a function $f : (0,1) \to (1,\infty)$ satisfies the condition*

$$f(pq) = f(p)f(q) \tag{27}$$

for all $p,q \in (0,1)$. There exists $c > 0$ such that $f(p) = p^{-c}$ for all $p \in (0,1)$.

Proof Equation (27) is one of the four fundamental Cauchy equations, which can be easily reduced to each other. For example, introducing a new function $g : (0,\infty) \to (0,\infty)$ by $g(u) := \ln f(e^{-u})$ and new variables $x,y \in (0,\infty)$ by $x := -\ln p$ and $y := -\ln q$, we transform (27) to the most standard Cauchy equation $g(x+y) = g(x)+g(y)$. By Darboux's theorem (see, e.g., Aczél, 1966, Section 2.1, Theorem 1(b)), $g(x) = cx$ for all $x > 0$, that is, $f(p) = p^{-c}$ for all $p \in (0,1)$. ■

The function f is defined on $(0,1)$ since we assume that in real life no bookmaker will assign a subjective probability of exactly 0 or 1 to an event on which he accepts bets. It would be irrational for the bookmaker to have $f(p) \le 1$ for some p, so $f : (0,1) \to (1,\infty)$. To justify the requirement (27), we assume that the bookmaker offers not only "single" but also "double" bets (Wikipedia, 2009). If there are two events with quoted odds a and b that the bookmaker considers independent, his quoted odds on the conjunction of the two events will be ab. If the probabilities of the two events are p and q, respectively, the probability of their conjunction will be pq. Therefore, we have (27).

Theorem 4 provides a justification of Khutsishvili's formula (3): we just assume that the bookmaker applies the same function f to all three probabilities p_1, p_2, and p_3. If $f(p) = p^{-c}$, we have $p_i = a_i^{-\gamma}$, where $\gamma = 1/c$ and $i = 1,2,3$, and γ can be found from the requirement $p_1 + p_2 + p_3 = 1$.

An important advantage of (3) over (26) is that (3) does not impose any upper limits on the overround that the bookmaker may charge (Khutsishvili, 2009). If the game has n possible outcomes ($n = 3$ for football and $n = 2$ for tennis) and the bookmaker uses $f(p) = p^{-c}$, the overround is

$$\sum_{i=1}^{n} a_i^{-1} - 1 = \sum_{i=1}^{n} p_i^c - 1$$

and so continuously changes between -1 and $n-1$ as c ranges over $(0,\infty)$ (in practice, the overround is usually positive, and so $c \in (0,1)$). Even for $n=2$, the upper bound of 1 is too large to be considered a limitation. The situation with (26) is very different: upper bounding the numerator of (26) by 1 and replacing the denominator by $1+o$, where o is the overround, we obtain $p_i < \frac{1}{1+o}$ for all i, and so $o < \min_i p_i^{-1} - 1$; this limitation on o is restrictive when one of the p_i is close to 1.

An interesting phenomenon in racetrack betting, known since Griffith (1949), is that favourites are usually underbet while longshots are overbet (see, e.g., Snowberg and Wolfers, 2007, for a recent survey and analysis). Khutsishvili's formula (3) can be regarded as a way of correcting this "favourite-longshot bias": when a_i is large (the outcome i is a longshot), (3) slashes $1/a_i$ when computing p_i more than (26) does.

Appendix C. Comparison with Other Prediction Algorithms

Other popular algorithms for prediction with expert advice that could be used instead of Algorithm 1 in our empirical studies reported in Section 3 are, among others, the Weighted Average Algorithm (WdAA, proposed by Kivinen and Warmuth, 1999), the weak aggregating algorithm (WkAA, proposed independently by Kalnishkan and Vyugin, 2008, and Cesa-Bianchi and Lugosi, 2006, Theorem 2.3; we are using Kalnishkan and Vyugin's name), and the Hedge algorithm (HA, proposed by Freund and Schapire, 1997). In this appendix we pay most attention to the WdAA since neither WkAA nor HA satisfy bounds of the form (2). (The reader can consult Vovk and Zhdanov, 2008b, for details of experiments with the latter two algorithms and formula (26) used for extracting probabilities from the quoted betting odds.) We also briefly discuss three more naive algorithms.

The Weighted Average Algorithm is very similar to the strong aggregating algorithm (SAA) used in this paper: the WdAA maintains the same weights for the experts as the SAA, and the only difference is that the WdAA merges the experts' predictions by averaging them according to their weights, whereas the SAA uses a more complicated "minimax optimal" merging scheme (given by (19) for the Brier game). The performance guarantee for the WdAA applied to the Brier game is weaker than the optimal (1), but of course this does not mean that its empirical performance is necessarily worse than that of the SAA (i.e., Algorithm 1). Figures 5 and 6 show the performance of this algorithm, in the same format as before (see Figures 1 and 3). We can see that for the football data the maximal difference between the cumulative loss of the WdAA and the cumulative loss of the best expert is slightly larger than that for Algorithm 1 but still well within the optimal bound $\ln K$ given by (1). For the tennis data the maximal difference is almost twice as large as for Algorithm 1, violating the optimal bound $\ln K$.

In its most basic form (Kivinen and Warmuth, 1999, the beginning of Section 6), the WdAA works in the following protocol. At each step each expert, Learner, and Reality choose an element of the unit ball in $\mathbb{R}^n$, and the loss function is the squared distance between the decision (Learner's or an expert's move) and the observation (Reality's move). This covers the Brier game with $\Omega = \{1,\dots,n\}$, each observation $\omega \in \Omega$ represented as the vector $(\delta_\omega\{1\},\dots,\delta_\omega\{n\})$, and each decision $\gamma \in \mathcal{P}(\Omega)$ represented as the vector $(\gamma\{1\},\dots,\gamma\{n\})$. However, in the Brier game the decision makers' moves are known to belong to the simplex $\{(u_1,\dots,u_n) \in [0,\infty)^n \mid \sum_{i=1}^n u_i = 1\}$, and Reality's move is known to be one of the vertices of this simplex. Therefore, we can optimize the ball radius by considering the smallest ball containing the simplex rather than the unit ball. This is what we did for the results reported here (although the results reported in the conference version of this paper, Vovk and Zhdanov, 2008a, are for the WdAA applied to the unit ball in $\mathbb{R}^n$). The

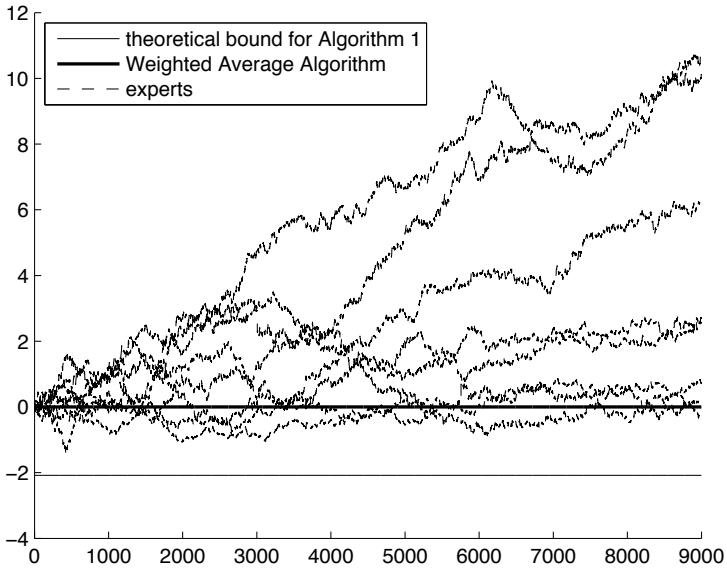


Figure 5: The difference between the cumulative loss of each of the 8 bookmakers and of the Weighted Average Algorithm (WdAA) on the football data. The chosen value of the parameter $c = 1/\eta$ for the WdAA, $c := 16/3$, minimizes its theoretical loss bound. The theoretical lower bound $-\ln 8 \approx -2.0794$ for Algorithm 1 is also shown (the theoretical lower bound for the WdAA, -11.0904, can be extracted from Table 3 below).

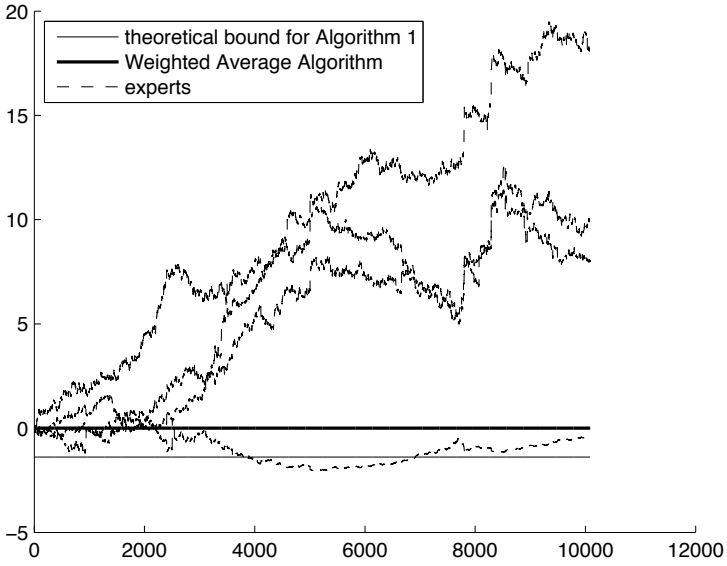


Figure 6: The difference between the cumulative loss of each of the 4 bookmakers and of the WdAA for $c := 4$ on the tennis data.

Algorithm	Maximal difference	Theoretical bound
Algorithm 1	1.2318	2.0794
WdAA ($c = 16/3$)	1.4076	11.0904
WdAA ($c = 1$)	1.2255	none

Table 3: The maximal difference between the loss of each algorithm in the selected set and the loss of the best expert for the football data (second column); the theoretical upper bound on this difference (third column).

radius of the smallest ball is

$$R := \sqrt{1 - \frac{1}{n}} \approx \begin{cases} 0.8165 & \text{if } n = 3 \\ 0.7071 & \text{if } n = 2 \\ 1 & \text{if } n \text{ is large.} \end{cases}$$

As described in Kivinen and Warmuth (1999), the WdAA is parameterized by $c := 1/\eta$ instead of η, and the optimal value of c is $c = 8R^2$, leading to the guaranteed loss bound

$$L_N \le \min_{k=1,\dots,K} L_N^k + 8R^2 \ln K$$

for all $N = 1, 2, \dots$ (see Kivinen and Warmuth, 1999, Section 6). This is significantly looser than the bound (1) for Algorithm 1.

The values $c = 16/3$ and $c = 4$ used in Figures 5 and 6, respectively, are obtained by minimizing the WdAA's performance guarantee, but minimizing a loose bound might not be such a good idea. Figure 7 shows the maximal difference

$$\max_{N=1,\dots,8999} \left(L_N(c) - \min_{k=1,\dots,8} L_N^k \right), \tag{28}$$

where $L_N(c)$ is the loss of the WdAA with parameter c on the football data over the first N steps and L_N^k is the analogous loss of the kth expert, as a function of c. Similarly, Figure 8 shows the maximal difference

$$\max_{N=1,\dots,10087} \left(L_N(c) - \min_{k=1,\dots,4} L_N^k \right) \tag{29}$$

for the tennis data. And indeed, in both cases the value of c minimizing the empirical loss is far from the value minimizing the bound; as could be expected, the empirical optimal value for the WdAA is not so different from the optimal value for Algorithm 1. The following two figures, 9 and 10, demonstrate that there is no such anomaly for Algorithm 1.

Figures 11 and 12 show the behaviour of the WdAA for the value of parameter $c = 1$, that is, $\eta = 1$, that is optimal for Algorithm 1. They look remarkably similar to Figures 1 and 3, respectively.

Precise numbers associated with the figures referred to above are given in Tables 3 and 4: the second column gives the maximal differences (28) and (29), respectively. The third column gives the theoretical upper bound on the maximal difference (i.e., the optimal value of A in (2), if available).

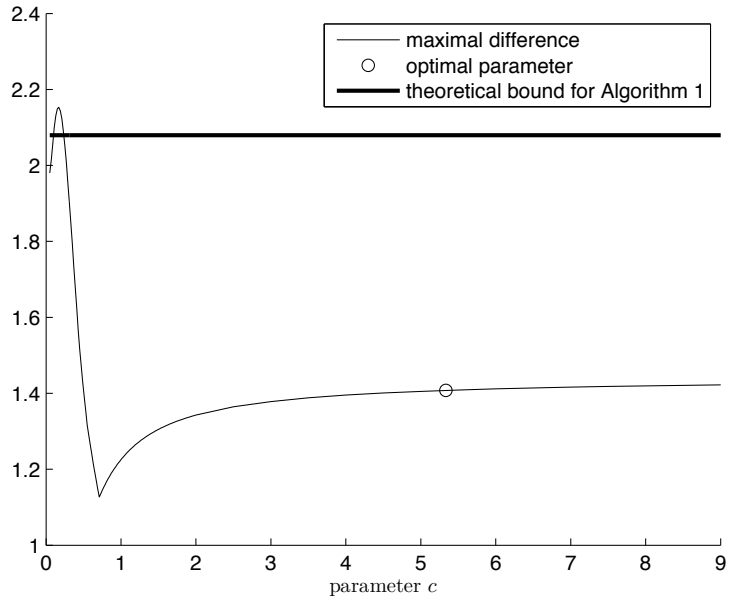


Figure 7: The maximal difference (28) for the WdAA as function of the parameter c on the football data. The theoretical guarantee $\ln 8$ for the maximal difference for Algorithm 1 is also shown (the theoretical guarantee for the WdAA, 11.0904, is given in Table 3).

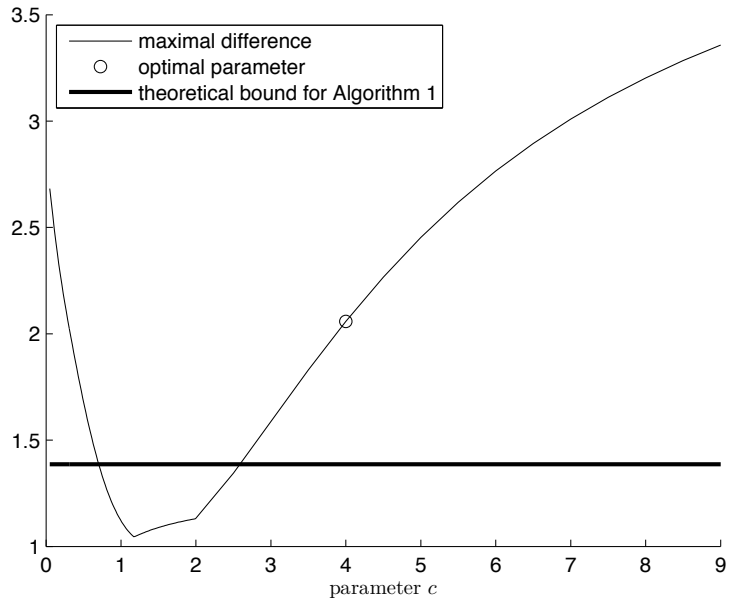


Figure 8: The maximal difference (29) for the WdAA as function of the parameter c on the tennis data. The theoretical bound for the WdAA is 5.5452 (see Table 4).

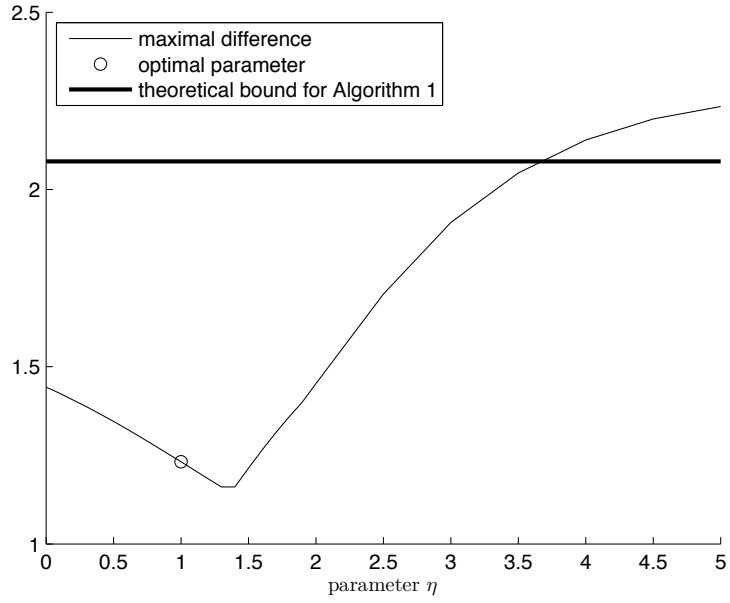


Figure 9: The maximal difference ((28) with η in place of c) for Algorithm 1 as function of the parameter η on the football data.

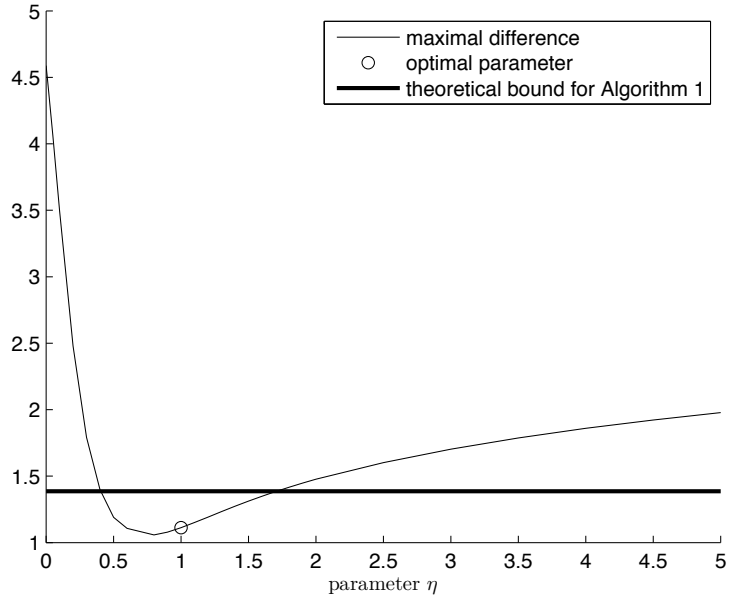


Figure 10: The maximal difference ((29) with η in place of c) for Algorithm 1 as function of the parameter η on the tennis data.

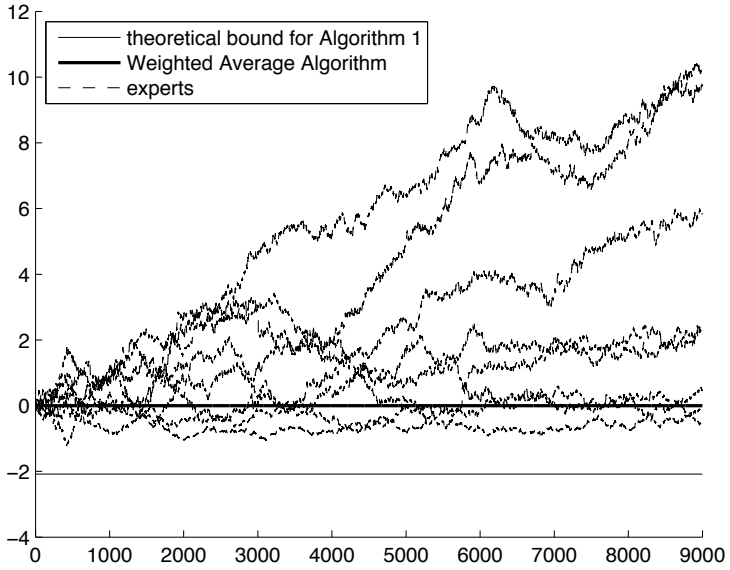


Figure 11: The difference between the cumulative loss of each of the 8 bookmakers and of the WdAA on the football data for $c = 1$ (the value of parameter minimizing the theoretical performance guarantee for Algorithm 1).

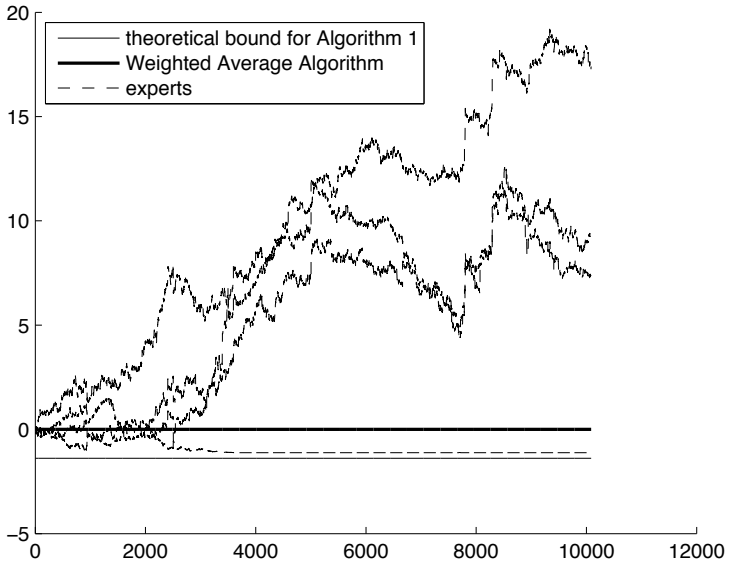


Figure 12: The difference between the cumulative loss of each of the 4 bookmakers and of the WdAA for $c = 1$ on the tennis data.

Algorithm	Maximal difference	Theoretical bound
Algorithm 1	1.1119	1.3863
WdAA ($c = 4$)	2.0583	5.5452
WdAA ($c = 1$)	1.1207	none

Table 4: The maximal difference between the loss of each algorithm in the selected set and the loss of the best expert for the tennis data (second column); the theoretical upper bound on this difference (third column).

The following two algorithms, the weak aggregating algorithm (WkAA) and the Hedge algorithm (HA), make increasingly weaker assumptions about the prediction game being played. Algorithm 1 computes the experts' weights taking full account of the degree of convexity of the loss function and uses a minimax optimal substitution function. Not surprisingly, it leads to the optimal loss bound of the form (2). The WdAA computes the experts' weights in the same way, but uses a suboptimal substitution function; this naturally leads to a suboptimal loss bound. The WkAA "does not know" that the loss function is strictly convex; it computes the experts' weights in a way that leads to decent results for all convex functions. The WkAA uses the same substitution function as the WdAA, but this appears less important than the way it computes the weights. The HA "knows" even less: it does not even know that its and the experts' performance is measured using a loss function. At each step the HA decides which expert it is going to follow, and at the end of the step it is only told the losses suffered by all experts. Both WkAA and HA depend on a parameter, which is denoted c in the case of WkAA and β in the case of HA; the ranges of the parameters are $c \in (0,\infty)$ and $\beta \in [0,1)$. The loss bounds that we give below assume that the loss function takes values in the interval $[0,L]$, in the case of the WkAA, and that the losses are chosen from $[0,L]$, in the case of HA, where L is a known constant. In the case of the Brier loss function, $L = 2$.

In the notation of (1), a simple loss bound for the WkAA is

$$L_N \le \min_{k=1,\dots,K} L_N^k + 2L\sqrt{N \ln K} \tag{30}$$

(Kalnishkan and Vyugin, 2008, Corollary 14); this is quite different from (1) as the "regret term" $2L\sqrt{N \ln K}$ in (30) depends on N. This bound is guaranteed for $c = \sqrt{\ln K}/L$. For $c = \sqrt{8 \ln K}/L$, Cesa-Bianchi and Lugosi (2006, Theorem 2.3) prove the stronger bound

$$L_N \le \min_{k=1,\dots,K} L_N^k + L\sqrt{2N \ln K} + L\sqrt{\frac{\ln K}{8}}.$$

The performance of the WkAA on our data sets is significantly worse than that of the WdAA with $c = 1$: the maximal difference (28)–(29) does not exceed $\ln K$ for all reasonable values of c in the case of football but only for a very narrow range of c (which is far from both Kalnishkan and Vyugin's $\sqrt{\ln K}/2$ and Cesa-Bianchi and Lugosi's $\sqrt{8 \ln K}/2$) in the case of tennis. Moreover, the WkAA violates the bound for Algorithm 1 for all reasonable values of c on some natural subsets of the football data set: for example, when prediction starts from the second (2006/2007) season. Nothing similar happens for the WdAA with $c = 1$ on our data sets.

The loss bound for the HA is

$$\mathbb{E}L_N \le \frac{L_N^* \ln \frac{1}{\beta} + L \ln K}{1-\beta} \tag{31}$$

(Freund and Schapire, 1997, Theorem 2), where $\mathbb{E}L_N$ stands for Learner's expected loss (the HA is a randomized algorithm) and L_N^* stands for $\min_{k=1,\dots,K} L_N^k$. In the same framework, the strong aggregating algorithm attains the stronger bound

$$\mathbb{E}L_N \le \frac{L_N^* \ln \frac{1}{\beta} + L \ln K}{K \ln \frac{K}{K+\beta-1}} \tag{32}$$

(Vovk, 1998, Example 7). Of course, the SAA applied to the HA framework (as described above, with no loss function) is very different from Algorithm 1, which is the SAA applied to the Brier game; we refer to the former algorithm as SAA-HA. Figure 13 shows the ratio of the right-hand side of (32) to the right-hand side of (31) as function of β.

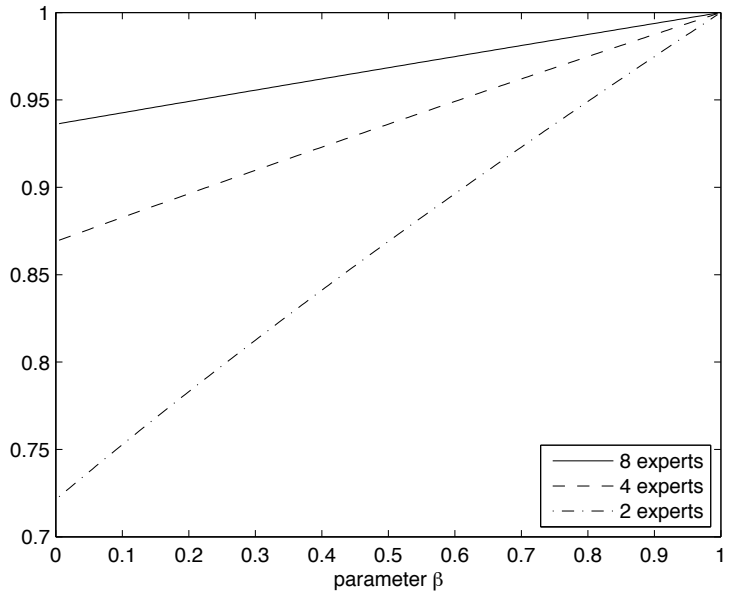


Figure 13: The relative performance of the HA and SAA-HA for various numbers of experts as function of parameter β.

The losses suffered by the HA and the SAA-HA on our data sets are very close and violate Algorithm 1's regret term $\ln K$ for all values of β. It is interesting that, for both football and tennis data, the loss of the HA is almost minimized by setting its parameter β to 0 (the qualification "almost" is necessary only in the case of the tennis data). The HA with $\beta = 0$ coincides with the Follow the Leader Algorithm (FLA), which chooses the same decision as the best (with the smallest loss up to now) expert; if there are several best experts (which almost never happens after the first step), their predictions are averaged with equal weights. Standard examples (see, e.g., Cesa-Bianchi

and Lugosi, 2006, Section 4.3) show that this algorithm (unlike its version Follow the Perturbed Leader) can fail badly on some data sequences. Its empirical performance on the football data set is not so bad: it violates the loss bound for Algorithm 1 only slightly; however, on the tennis data set the bound is violated badly.

The decent performance of the Follow the Leader Algorithm on the football data set suggests checking the empirical performance of other similarly naive algorithms, such as the following two. The *Simple Average Algorithm*'s decision is defined as the arithmetic mean of the experts' decisions (with equal weights). The *Bayes Mixture Algorithm* (BMA) is the strong aggregating algorithm applied to the log loss function; this algorithm is in fact optimal, but not for the Brier loss function. The BMA has a very simple description (Cesa-Bianchi and Lugosi, 2006, Section 9.2), and was studied from the point of view of prediction with expert advice already in DeSantis et al. (1988).

We have found that none of the three naive algorithms perform consistently poorly, but they always fail badly on some natural part of our data sets. The advantage of the more sophisticated algorithms having strong performance guarantees is that there is no danger of catastrophic performance on any data set.

References

János Aczél. *Lectures on Functional Equations and their Applications*. Academic Press, New York, 1966.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78:1–3, 1950.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, Cambridge, England, 2006.

Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the Association for Computing Machinery*, 44:427–485, 1997.

A. Philip Dawid. Probability forecasting. In Samuel Kotz, Norman L. Johnson, and Campbell B. Read, editors, *Encyclopedia of Statistical Sciences*, volume 7, pages 210–218. Wiley, New York, 1986.

Alfredo DeSantis, George Markowsky, and Mark N. Wegman. Learning probabilistic prediction functions. In *Proceedings of the Twenty Ninth Annual IEEE Symposium on Foundations of Computer Science*, pages 110–119, Los Alamitos, CA, 1988. IEEE Computer Society.

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*, 55:119–139, 1997.

Richard M. Griffith. Odds adjustments by American horse-race bettors. *American Journal of Psychology*, 62:290–294, 1949.

Godfrey H. Hardy, John E. Littlewood, and George Pólya. *Inequalities*. Cambridge University Press, Cambridge, England, second edition, 1952.

David Haussler, Jyrki Kivinen, and Manfred K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44:1906–1925, 1998.

Yuri Kalnishkan and Michael V. Vyugin. The weak aggregating algorithm and weak mixability. *Journal of Computer and System Sciences*, 74:1228–1244, 2008. Special Issue devoted to COLT 2005.

Victor Khutsishvili. Personal communication. E-mail exchanges (from 27 November 2008), 2009.

Jyrki Kivinen and Manfred K. Warmuth. Averaging expert predictions. In Paul Fischer and Hans U. Simon, editors, *Proceedings of the Fourth European Conference on Computational Learning Theory*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 153–167, Berlin, 1999. Springer.

Nick Littlestone and Manfred K. Warmuth. The Weighted Majority Algorithm. *Information and Computation*, 108:212–261, 1994.

Erik Snowberg and Justin Wolfers. Explaining the favorite-longshot bias: Is it risk-love or misperceptions? Available on-line at `http://bpp.wharton.upenn.edu/jwolfers/` (accessed on 2 November 2009), November 2007.

John A. Thorpe. *Elementary Topics in Differential Geometry*. Springer, New York, 1979.

Vladimir Vovk. Aggregating strategies. In Mark Fulk and John Case, editors, *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383, San Mateo, CA, 1990. Morgan Kaufmann.

Vladimir Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56:153–173, 1998.

Vladimir Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35:247–282, 1999.

Vladimir Vovk. Competitive on-line statistics. *International Statistical Review*, 69:213–248, 2001.

Vladimir Vovk and Fedor Zhdanov. Prediction with expert advice for the Brier game. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the Twenty Fifth International Conference on Machine Learning*, pages 1104–1111, New York, 2008a. ACM.

Vladimir Vovk and Fedor Zhdanov. Prediction with expert advice for the Brier game. Technical Report `arXiv:0708.2502v2` [cs.LG], `arXiv.org` e-Print archive, June 2008b.

Wikipedia. Glossary of bets offered by UK bookmakers — Wikipedia, The Free Encyclopedia, 2009. Accessed on 2 November.

Journal of Machine Learning Research 10 (2009) 2473-2505 Submitted 12/08; Revised 8/09; Published 11/09

Bi-Level Path Following for Cross Validated Solution of Kernel Quantile Regression

Saharon Rosset SAHARON@POST.TAU.AC.IL
Department of Statistics
The Raymond and Beverly Sackler School of Mathemetical Sciences
Tel Aviv University
Tel Aviv, Israel

Editor: Ingo Steinwart

Abstract

We show how to follow the path of *cross validated* solutions to families of regularized optimization problems, defined by a combination of a parameterized loss function and a regularization term. A primary example is kernel quantile regression, where the parameter of the loss function is the quantile being estimated. Even though the bi-level optimization problem we encounter for every quantile is non-convex, the manner in which the optimal cross-validated solution evolves with the parameter of the loss function allows tracking of this solution. We prove this property, construct the resulting algorithm, and demonstrate it on real and artificial data. This algorithm allows us to efficiently solve the whole family of bi-level problems. We show how it can be extended to cover other modeling problems, like support vector regression, and alternative in-sample model selection approaches.[1]

1. Introduction

In the standard predictive modeling setting, we are given a *training sample* of n examples $\{\mathbf{x}_1, y_1\}, ..., \{\mathbf{x}_n, y_n\}$ drawn i.i.d from a joint distribution $P(X,Y)$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ for regression, $y_i \in \{0,1\}$ for two-class classification. We aim to employ these data to build a model $\hat{Y} = \hat{f}(X)$ to describe the relationship between X and Y, and later use it to predict the value of Y given new X values. This is often done by defining a family of models $\mathcal{F}$ and finding (exactly or approximately) the model $f \in \mathcal{F}$ which minimizes an *empirical loss function*: $\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))$. Examples of such algorithms include linear and logistic regression, empirical risk minimization in classification and others.

If $\mathcal{F}$ is complex, it is often desirable to add *regularization* to control model complexity and overfitting. The generic regularized optimization problem can be written as:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f) \ ,$$

where $J(f)$ is an appropriate model complexity penalty and λ is the regularization parameter. Given a loss and a penalty, selection of a good value of λ is a *model selection* problem. Popular approaches that can be formulated as regularized optimization problems include all versions of support vector

1. A short version of this paper appeared at ICML 2008 (Rosset, 2008).

machines, ridge regression, the Lasso and many others. For an overview of predictive modeling, regularized optimization and the algorithms mentioned above, see for example Hastie et al. (2001).

In this paper we are interested in a specific setup where we have a family of regularized optimization problems, defined by a parameterized loss function and a regularization term. A major motivating example for this setting is regularized quantile regression (Koenker, 2005). In regularized linear quantile regression we take the family $\mathcal{F}$ to be all linear combinations characterized by a coefficient vector $\beta \in \mathbb{R}^p$ and the modeling problem is

$$\hat{\beta}(\tau,\lambda) = \arg\min_{\beta} \sum_{i=1}^{n} L_\tau(y_i - \beta^T \mathbf{x}_i) + \lambda \|\beta\|_q^q \ , \ 0 < \tau < 1, \ 0 \leq \lambda < \infty \ , \tag{1}$$

where L_τ, the parameterized quantile loss function, has the form:

$$L_\tau(r) = \begin{cases} r\tau & r \geq 0 \\ -r(1-\tau) & r < 0 \end{cases} \ ,$$

and is termed τ-*quantile loss* because its population optimizer is the appropriate quantile (Koenker, 2005):

$$\arg\min_{c} E(L_\tau(Y-c)|X=\mathbf{x}) = \text{quantile } \tau \text{ of } P(Y|X=\mathbf{x}) \ . \tag{2}$$

Because quantile loss has this optimizer, the solution of the quantile regression problems for the whole range $0 < \tau < 1$ has often been advocated as an approach to estimating the full conditional probability of $P(Y|X)$ (Koenker, 2005; Perlich et al., 2007). Much of the interesting information about the behavior of $Y|X$ may lie in the details of this conditional distribution, and if it is not *nicely behaved* (i.i.d Gaussian noise being the most commonly used concept of nice behavior), just estimating a conditional mean or median is often not sufficient to properly understand and model the mechanisms generating Y. The importance of estimating a complete conditional distribution, and not just a central quantity like the conditional mean, has long been noted and addressed in various communities, like econometrics, education and finance (Koenker, 2005; Buchinsky, 1994; Eide and Showalter, 1998). There has been a surge of interest in the machine learning community in conditional quantile estimation in recent years, including theoretical analyses of consistency in quantile estimation and connections with support vector machines (Steinwart and Christmann, 2008; Christmann and Steinwart, 2008); methodological work on algorithms for quantile regression and their performance (Meinshausen, 2006; Takeuchi et al., 2006; Mease et al., 2007); and work on practical uses of extreme quantile estimation for data mining applications Perlich et al. (2007). Figure 1 shows a graphical representation of L_τ for several values of τ, and a demonstration of the conditional quantile curves in a univariate regression setting, where the linear model is correct for the median, but the noise has a non-homoscedastic distribution.

On the penalty side, we typically use the ℓ_q norm of the parameters with $q \in \{1,2\}$. Adding a penalty can be thought of as shrinkage, complexity control or putting a prior to express our expectation that the β's should be small.

As has been noted in the literature (Rosset and Zhu, 2007; Hastie et al., 2004; Li et al., 2007; Takeuchi et al., 2009) if $q \in \{1,2\}$ and if we fix $\tau = \tau_0$, we can devise *path following* (AKA parametric programming) algorithms to efficiently generate the 1-dimensional curve of solutions $\{\hat{\beta}(\tau_0,\lambda) \ : \ 0 \leq \lambda < \infty\}$. Although it has not been explicitly noted by most of these authors (a notable exception being Takeuchi et al. 2009), it naturally follows that similar algorithms exist for the case that we fix $\lambda = \lambda_0$ and are interested in generating the curve $\{\hat{\beta}(\tau,\lambda_0) \ : \ 0 < \tau < 1\}$.

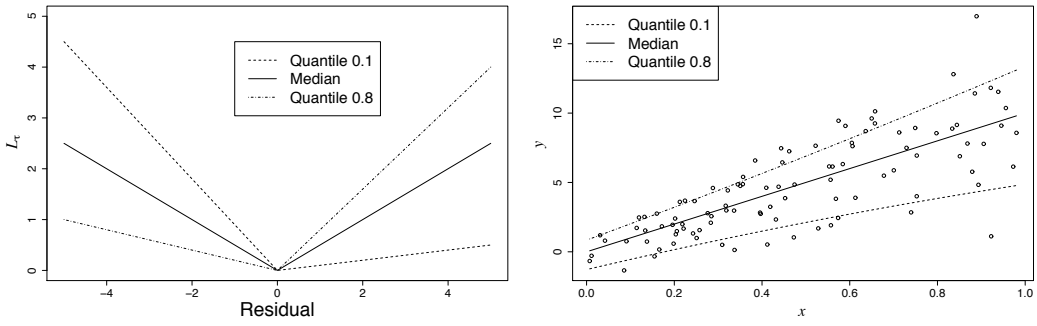


Figure 1: Quantile loss function for some values of τ (left) and an example where the median of Y is linear in X but the quantiles of $P(Y|X)$ are not because the noise is not identically distributed (right).

In addition to parameterized quantile regression, there are other modeling problems in the literature which combine a parameterized loss function problem with the existence of efficient path following algorithms. These include :

1. Support vector regression (SVR, Smola and Schölkopf 2004, see Gunther and Zhu 2005 for path following algorithm) with ℓ_1 or ℓ_2 regularization, where the parameter ε determines the width of the *don't care* region around 0.

2. Weighted support vector machines, where the parameter of the loss function corresponds to reweighting the hinge loss differentially for the two classes, for example as a means for deriving accurate probability estimates (as recently suggested by Wang et al. 2008).

3. Huberized Lasso (Rosset and Zhu, 2007) with ℓ_1 regularization, where *huberizing* adds robustness to the traditional squared error loss, with a tunable parameter.

An important extension of the ℓ_2-regularized optimization problem is to *non-linear* fitting through kernel embedding (Schölkopf and Smola, 2002). The kernelized version of Problem (1) is

$$\hat{f}(\tau,\lambda) = \arg\min_f \sum_i L_\tau(y_i - f(\mathbf{x}_i)) + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}_K}\,, \tag{3}$$

where $\|\cdot\|_{\mathcal{H}_K}$ is the norm induced by the positive-definite kernel K in the Reproducing Kernel Hilbert Space (RKHS) it generates. The well known *representer theorem* (Kimeldorf and Wahba, 1971) implies that the solution of Problem (3) lies in a low dimensional subspace spanned by the representer functions $\{K(\cdot,\mathbf{x}_i), i \in 1,...,n\}$. Following the ideas of Hastie et al. (2004) for the support vector machine, Li et al. (2007) have shown that the λ-path of solutions to Problem (3) when τ is fixed can also be efficiently generated. A similar approach was independently developed by Takeuchi et al. (2009).

It is important to note the difference in the roles of the two parameters τ, λ. The former defines a family of loss functions, in our case leading to estimation of different quantiles. Thus we would typically want to build and use a model for every value of τ. The latter is a regularization parameter, controlling model complexity with the aim of generating a better model and avoiding overfitting,

and is not part of the prediction objective (at least as long as we avoid the Bayesian view). We would therefore typically want to generate a set of models $\beta^*(\tau)$ (or $f^*(\tau)$ in the kernel case), by selecting a good regularization parameter $\lambda^*(\tau)$ for every value of τ, thus obtaining a family of good models for estimating the range of conditional quantiles, and consequently the whole conditional distribution.

This problem, of model selection to find a good regularization parameter, is often handled through *cross-validation*. In its simplest form, cross-validation entails having a second, independent set of data $\{\tilde{\mathbf{x}}_i, \tilde{y}_i\}_{i=1}^N$ (often referred to as a *validation set*), which is used to evaluate the performance of the models and select a good regularization parameter. For a fixed τ, we can write our model selection problem as a *Bi-level programming* extension of Problems (1) and (3), where $f^*(\tau) = \hat{f}(\tau, \lambda^*)$ and λ^* solves

$$\min_{\lambda} \quad \sum_{i=1}^{N} L_{\text{cv}}(\tilde{y}_i, \hat{f}(\tau,\lambda)^T \tilde{\mathbf{x}}_i) \qquad (4)$$
$$\text{s.t. } \hat{f}(\tau,\lambda) \text{ solves Problem (3)} ,$$

where L_{CV} is the cross validation loss function (the bi-level formulation for Problem (1) would be identical, with $\hat{\beta}$ replacing $\hat{f}$). We will assume for now that $L_{\text{CV}} = L_\tau$, in order to evaluate the performance in estimating the τth quantile. The objective of this minimization problem is not convex as a function of λ. A similar non-convex optimization problem has been tackled by Kunapuli et al. (2008) for the support vector machine, which is very similar to quantile regression from an optimization perspective (piecewise linear objective with quadratic penalty). The fundamental difference between their setting and ours is that they had a single bi-level optimization problem, while we have a family of such problems, parameterized by τ. This allows us to take advantage of internal structure to solve the bi-level problem for all values of τ *simultaneously* (or more accurately, in one run of our algorithm).

The concept of a parameterized family of bi-level regularized quantile regression problems is demonstrated in Figure 2, where we see the cross-validation curves of the objective of (4) as a function of λ for several values of τ on the same data set. As we can see, the optimal level of regularization varies with the quantile, and correct choice of the regularization parameter can have a significant effect on the success of our quantile prediction model.

The main goal of this paper is to devise algorithms for following the bi-level optimal solution path $f^*(\tau)$ as a function of τ, and demonstrate their practicality. Our algorithms are based on extensions and generalizations of some of the ideas underlying the path following algorithms for 1-dimensional paths on convex problems (Hastie et al., 2004; Li et al., 2007; Rosset and Zhu, 2007). We concentrate our attention on the quantile regression case (both kernelized and linear), as one where the parameterized-loss problem is well motivated and historically useful, but we also discuss the similarities and differences in algorithms for the other examples we mentioned above. We show that this non-convex family of bi-level programs can be solved exactly, as the optimum among the solutions of $O(n+N)$ standard (convex) path-following problems, with some additional twists. This result is based on a characterization of the evolution of the solution path $\hat{f}(\tau,\cdot)$ as τ varies, and on an understanding of the properties of optimal solutions of the bi-level problem, which can only occur at a limited set of points. We combine these insights to formulate an actual algorithm for solving this family of bi-level programs via path-following. However, this algorithm carries a heavy computational burden. The question of whether it is practical from a computational perspective depends on

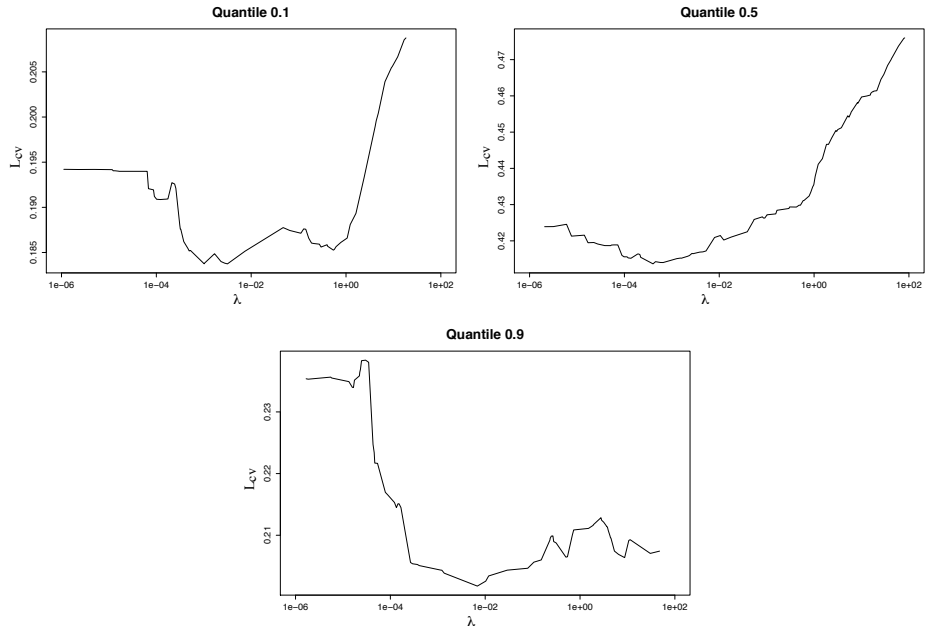


Figure 2: Estimated prediction error curves of Kernel Quantile Regression for some quantiles on one data set. The errors are shown as a function of the regularization parameter λ.

the properties of the modeling problems at hand, and may also benefit greatly from computational tricks and optimization shortcuts which are not the focus of this paper. We demonstrate its ability to successfully generate the complete set of cross-validated solutions on some illuminating simulation problems and on two medium-size real-life data-sets.

The rest of this paper is organized as follows. In Section 2 we discuss the properties of the quantile regression solution paths $\hat{f}(\tau,\lambda)$ and their evolution as τ changes. We then discuss in Section 3 the properties of the bi-level optimization Problem (4) and demonstrate that the solutions change predictably with τ. This is because the optimal solution always corresponds to a situation where either one of the validation points is crossing the non-differentiability *elbow* in the cross validation loss L_{CV}, or the regularization path is going thorough a *knot* in its piecewise linear change. However, due to the non-convexity of the problem, the solutions occasionally "jump" from one such point to another. It turns out that to follow this jumpy behavior we need to follow, not one path of solutions, but about $N+n$ of them, corresponding to all possible candidates for L_{CV} optimizers. Bringing together all our insights leads us to formulate an algorithm in Section 4, which allows us to follow the path of solutions $\{f^*(\tau)\ ,\ 0<\tau<1\}$ and only requires following a large but manageable number of solution paths of problem (3) simultaneously. In Section 5 we discuss the extension of our methodology to other scenarios, including application of our methodology to SVR. We demonstrate our methods with a simulated and real data study in Section 6, where we show that our approach leads to model-selection that is more efficient than previous approaches, and illustrate the interesting behavior of KQR in practice.

2. Quantile Regression Solution Paths

We concentrate our discussion on the kernel quantile regression (KQR) formulation in (3), with the understanding that it subsumes the linear formulation (1) with ℓ_2 regularization by using the *linear* kernel $K(\mathbf{x},\tilde{\mathbf{x}}) = \mathbf{x}^\intercal\tilde{\mathbf{x}}$.

We briefly survey the results of Li et al. (2007) regarding the properties of $\hat{f}(\tau,\cdot)$, the optimal solution path of (3), with τ fixed. Similar results were independently generated by Takeuchi et al. (2009), who concentrate on the properties of $\hat{f}(\tau,\cdot)$ with λ fixed (as we elaborate below, these problems are in fact very similar). The representer theorem (Kimeldorf and Wahba, 1971) implies that the solution can be written as

$$\hat{f}(\tau,\lambda)(\mathbf{x}) = \frac{1}{\lambda}\left[\hat{\beta}_0 + \sum_{i=1}^{n}\hat{\theta}_i K(\mathbf{x},\mathbf{x}_i)\right] \quad . \tag{5}$$

For a proposed solution $f(\mathbf{x})$ define:

- $\mathcal{E} = \{i : y_i - f(\mathbf{x}_i) = 0\}$ (points on *elbow* of L_τ)
- $\mathcal{L} = \{i : y_i - f(\mathbf{x}_i) < 0\}$ (left of elbow)
- $\mathcal{R} = \{i : y_i - f(\mathbf{x}_i) > 0\}$ (right of elbow).

Then Li et al. (2007) show that the Karush-Kuhn-Tucker (KKT) conditions for optimality of a solution $\hat{f}(\tau,\lambda)$ of problem (3) can be phrased as

- $i \in \mathcal{E} \;\Rightarrow\; -(1-\tau) \le \hat{\theta}_i \le \tau$
- $i \in \mathcal{L} \;\Rightarrow\; \hat{\theta}_i = -(1-\tau)$
- $i \in \mathcal{R} \;\Rightarrow\; \hat{\theta}_i = \tau$
- $\sum_i \hat{\theta}_i = 0$.

With some additional algebra, they show that for a fixed τ, there is a series of *knots*, $0 = \lambda_0 < \lambda_1 < \ldots < \lambda_m < \infty$ such that for $\lambda \ge \lambda_m$ we have $\hat{f}(\tau,\lambda) = \textit{constant}$ and for $\lambda_{k-1} < \lambda \le \lambda_k$ we have

$$\hat{f}(\tau,\lambda)(\mathbf{x}) = \frac{1}{\lambda}\left(\lambda_k \hat{f}(\tau,\lambda_k)(\mathbf{x}) + (\lambda - \lambda_k)h_k(\mathbf{x})\right) \ , \tag{6}$$

where $h_k(\mathbf{x}) = b_0^k + \sum_{i\in\mathcal{E}_k} b_i^k K(\mathbf{x},\mathbf{x}_i)$ can be thought of as the *direction* in which the solution is moving for the region $\lambda_{k-1} < \lambda \le \lambda_k$. The knots λ_k are points on the path where an observation passes between $\mathcal{E}$ and either $\mathcal{L}$ or $\mathcal{R}$, that is $\exists i \in \mathcal{E}$ such that exactly $\theta_i = \tau$ or $\theta_i = -(1-\tau)$. This observation may be either *entering* the elbow (if it was previously in $\mathcal{L}$ or $\mathcal{R}$), or *exiting* it (if it previously had $\theta_i \in (-(1-\tau),\tau)$).[2] These insights lead Li et al. (2007) to an algorithm for incrementally generating $\hat{f}(\tau,\lambda)$ as a function of λ for fixed τ, starting from $\lambda = \infty$ (where the solution only contains the intercept β_0).

2. It is clear that the definition of an observation as *entering* or *exiting* the elbow is arbitrary, since an observation which enters at λ_k when we are decreasing λ actually exits at λ_k if we choose to traverse the path while increasing λ. There is also a possibility of more than one observation making this transition at once. With general τ and points in general location, this event has probability zero. As we will see, in the course of our investigation of the paths we are bound to encounter such cases, and we will address this issue when it comes up.

Although Li et al. (2007) suggest it is a topic for further study, it is in fact a reasonably straight forward extension of their results to show that a similar scenario holds when we fix λ and allow τ only to change. As previously mentioned, this has been recognized and used by other authors, including Takeuchi et al. (2009) for quantile regression, and Wang et al. (2008) for weighted hinge loss. More interestingly, the same is also true when both τ, λ are changing together *along a straight line*, that is, a 1-dimensional subspace of the (τ,λ) space (this has been observed by Wang et al. (2006) for SVR, which is very similar from an optimization perspective). The following lemma makes this more general result concrete. The proof relies on a study of the KKT conditions in the spirit of Li et al. (2007) and the other references above, and we omit it.

Lemma 1 *Let $\tau(\lambda) = u\lambda + v$, and denote $\hat{f}(\lambda) = \hat{f}(\tau(\lambda),\lambda)$. Then in the range $\Gamma = \{\lambda \geq 0 :\ 0 < \tau(\lambda) < 1\}$ there exist* knots $\lambda_0 < ... < \lambda_m$ *such that for $\lambda_{k-1} < \lambda \leq \lambda_k$ we have:*

$$\hat{f}(\lambda)(\mathbf{x}) = \frac{1}{\lambda}\left(\lambda_k \hat{f}(\lambda_k)(\mathbf{x}) + (\lambda - \lambda_k) h_k(\mathbf{x})\right) ,$$

where $h_k(\mathbf{x}) = b_0^k + \sum_{i\in\mathcal{E}_k} b_i^k K(\mathbf{x},\mathbf{x}_i)$, and the direction $\mathbf{b}^k = \begin{pmatrix} b_0^k \\ \vdots \\ b_{|\mathcal{E}_k|}^k \end{pmatrix}$ is the solution of a set of linear equations with $|\mathcal{E}_k|+1$ unknowns:

$$\mathbf{A}^k \mathbf{b}^k = \begin{pmatrix} 0 \\ \mathbf{r}_{\mathcal{E}_k} \end{pmatrix}$$

with

$$\mathbf{A}^k = \begin{pmatrix} 0 & \mathbf{1}^\intercal \\ \mathbf{1} & \mathbf{K}_{\mathcal{E}_k} \end{pmatrix} ,$$

as defined in Li et al. (2007); and $r_j = y_j + u \cdot \left(\sum_{i\in\mathcal{R}_k} K(\mathbf{x}_j,\mathbf{x}_i) - \sum_{i\in\mathcal{L}_k} K(\mathbf{x}_j,\mathbf{x}_i)\right)$ for $j \in \mathcal{E}_k$.

Armed with this result, we next show the main result of this section: that the knots themselves move in a (piecewise) straight line as τ changes, and can therefore be *tracked* as τ and the regularization path change. Fix a quantile τ_0 and assume that λ_k is a knot in the λ-solution path for quantile τ_0. Further, let i_k be the observation that is passing in or out of the elbow at knot λ_k. Assume WLOG that $\hat{\theta}_{i_k}(\tau_0,\lambda_k) = \tau_0$, that is, it is on the boundary between $\mathcal{R}_k$ and $\mathcal{E}_k$. Let $\tilde{\mathbf{K}}_{\mathcal{E}_k}$ be the matrix $\mathbf{K}_{\mathcal{E}_k}$ with the i_k column removed, and $\tilde{\mathbf{b}}^k = \mathbf{b}^k$ with index i_k removed. Let $s_i = \sum_{j\in\mathcal{R}\cup\mathcal{L}\cup\{i_k\}} K(\mathbf{x}_i,\mathbf{x}_j)$ for $i \in \mathcal{E}_k$. Let $\mathbf{s}_{\mathcal{E}_k}$ be the vector of all these values.

Theorem 2 *Any knot λ_k moves linearly as τ changes. That is, there exists a constant c_k such that for quantile $\tau_0 + \delta$ there is a knot in the λ-solution path at $\lambda_k + c_k\delta$, for $\delta \in [-\varepsilon_k, \nu_k]$, a non-empty neighborhood of 0. c_k is determined through the solution of another set of $|\mathcal{E}_k|+1$ linear equations with $|\mathcal{E}_k|+1$ unknowns*

$$\mathbf{B}^k \begin{pmatrix} \tilde{\mathbf{b}}^k \\ c_k \end{pmatrix} = \begin{pmatrix} -(|\mathcal{R}| + |\mathcal{L}| + 1) \\ -\mathbf{s}_{\mathcal{E}_k} \end{pmatrix} ,$$

with

$$\mathbf{B}^k = \begin{pmatrix} 0 & \mathbf{1}^\intercal & 0 \\ \mathbf{1} & \tilde{\mathbf{K}}_{\mathcal{E}_k} & -\mathbf{y}_{\mathcal{E}_k} \end{pmatrix} .$$

And the fit at this knot progresses as

$$\hat{f}(\tau_0+\delta,\lambda_k+c_k\delta) = \frac{1}{\lambda_k+c_k\delta}\left(\lambda_k\hat{f}(\lambda_k,\tau_0)(\mathbf{x})+\delta h_k(\mathbf{x})\right) \tag{7}$$

$$h_k(\mathbf{x}) = \tilde{b}_0^k + \sum_{i\in\mathcal{E}_k-i_k}\tilde{b}_i^k K(\mathbf{x},\mathbf{x}_i) + \sum_{i\in\mathcal{L}\cup\mathcal{R}\cup\{i_k\}} K(\mathbf{x},\mathbf{x}_i)\ . \tag{8}$$

Proof For small δ, the *modified* knot should be characterized by $\theta_{i_k} = \tau_0+\delta$, and $\mathcal{L},\mathcal{R},\mathcal{E}$ remaining the same. If we can find a c_k such that this holds for quantile $\tau_0+\delta$ and $\lambda = \lambda_k + c_k\delta$, and also the KKT conditions are maintained, we have our knot for quantile $\tau_0+\delta$. Assume this can be accomplished by moving in a direction h_k as in Equations (7) and (8). For the KKT conditions to hold we need to maintain:

C1. $\hat{f}(\tau_0+\delta,\lambda_k+c_k\delta)(\mathbf{x}_i) = y_i\ ,\ \ \forall i\in\mathcal{E}$

C2. $\hat{\theta}_i(\tau_0+\delta,\lambda_k+c_k\delta) = \hat{\theta}_i(\tau_0,\lambda_k)+\delta\ ,\ \ \forall i\in\mathcal{L}\cup\mathcal{R}\cup\{i_k\}$

C3. $\sum_{i\in\mathcal{E}_k-\{i_k\}}\tilde{b}_i^k = -|\mathcal{L}\cup\mathcal{R}\cup\{i_k\}|$

where C1 maintains the observations in $\mathcal{E}$ at the elbow, C2 maintains the equality requirements on $\hat{\theta}$ for the observations in $\mathcal{L}\cup\mathcal{R}$ (and the one on the boundary), and C3 maintains the constraint that the $\hat{\theta}$'s sum to 0.

We can express C1 in terms of Equations (7) and (8) and condition C2:

$$\begin{aligned} & \hat{f}(\tau_0+\delta,\lambda_k+c_k\delta)(\mathbf{x}_i) = y_i\ ,\ \ \forall i\in\mathcal{E} \\ \Leftrightarrow \quad & h_k(\mathbf{x}_i) = \tilde{b}_0^k + \sum_{j\in\mathcal{E}_k-i_k}\tilde{b}_j^k K(\mathbf{x}_j,\mathbf{x}_i) + \sum_{j\in\mathcal{L}\cup\mathcal{R}\cup\{i_k\}} K(\mathbf{x}_j,\mathbf{x}_i) = c_k y_i\ ,\ \ \forall i\in\mathcal{E}\ , \end{aligned} \tag{9}$$

where the last term on the RHS of (9) accounts for the changes in the $\hat{\theta}_j$ which C2 implies for $j\in\mathcal{L}\cup\mathcal{R}\cup\{i_k\}$.

Now, by combining C3 and (9) into one set of equations in matrix notation, we get the result of the theorem:

$$\begin{pmatrix}\tilde{\mathbf{b}}^k \\ c_k\end{pmatrix} = \left(\mathbf{B}^k\right)^{-1}\begin{pmatrix}-(|\mathcal{R}|+|\mathcal{L}|+1) \\ -\mathbf{s}_{\mathcal{E}_k}\end{pmatrix}$$

and moving in the direction of the solution of this matrix equation as in (7,8) maintains the KKT conditions and the observation at the elbow, hence is a knot on the λ-solution path for quantile $\tau+\delta$ for every (small enough) δ.

■

This theorem tells us that we can in fact track the knots in the solution efficiently as τ changes. We still have to account for various types of *events* that can change the direction the knot is moving in. The value θ_i for a point in $\mathcal{E}_k-\{i_k\}$ can reach τ or $-(1-\tau)$, or a point in $\mathcal{L}\cup\mathcal{R}$ may reach the elbow $\mathcal{E}$. These events correspond to *knots crossings*, that is, the knot λ_k is encountering another knot (which is tracking the other event). There are also *knot birth* events, and *knots merge* events, which are possible but rare, and somewhat counter-intuitive. We defer the details of how these are identified and handled to the detailed algorithm description (Appendix A). When any of these events occurs, the set of knots has to be updated and their directions have to be re-calculated using Lemma 1, Theorem 2 and the new identity of the sets $\mathcal{E},\mathcal{L},\mathcal{R}$ and the observation i_k. This in essence allows us to map the whole 2-dimensional solution surface $\hat{f}(\tau,\lambda)$.

3. The Bi-level Optimization Problem

Our next task is to show how our ability to track the knots as τ changes allows us to track the solution of the bi-level optimization Problem (4) as τ changes. The key to this step is the following result.

Theorem 3 *When the cross validation loss is the quantile loss (i.e., $L_{CV} = L_\tau$), then any minimizer*[3] *of (4) is always either at a knot in the λ-path for this τ or a point where a validation observation crosses the elbow. In other words, one of the two following statements must hold:*

- λ^* *is a knot:* $\exists i \in \{1...n\}$ *s.t.* $\hat{f}(\tau, \lambda^*(\tau))(\mathbf{x}_i) = y_i$ *and* $\theta_i \in \{\tau, -(1-\tau)\}$, *or*
- λ^* *is a* validation crossing*:* $\exists i \in \{1...N\}$ *s.t.* $\hat{f}(\tau, \lambda^*(\tau))(\tilde{\mathbf{x}}_i) = \tilde{y}_i$

Proof Define $\tilde{\mathcal{L}}, \tilde{\mathcal{R}}$ in the obvious way, as the sets of validation observations with negative and positive residuals, respectively, for a given model. Fix τ, and consider the cross validation loss for a given value of λ:

$$\begin{aligned} L_{\text{cv}}(\lambda) &:= \sum_{i=1}^{N} L_{\text{CV}}(\tilde{y}_i, \hat{f}(\lambda)(\tilde{\mathbf{x}}_i)) = \sum_{i \in \tilde{\mathcal{L}}} (1-\tau)(\hat{f}(\lambda)(\tilde{\mathbf{x}}_i) - \tilde{y}_i) + \sum_{i \in \tilde{\mathcal{R}}} \tau \cdot (\tilde{y}_i - \hat{f}(\lambda)(\tilde{\mathbf{x}}_i)) = \\ &= \tau \sum_{i \in \tilde{\mathcal{R}}} y_i - (1-\tau) \sum_{i \in \tilde{\mathcal{L}}} y_i - \tau \sum_{i \in \tilde{\mathcal{R}}} \hat{f}(\lambda_k)(\tilde{\mathbf{x}}_i) + (1-\tau) \sum_{i \in \tilde{\mathcal{L}}} \hat{f}(\lambda_k)(\tilde{\mathbf{x}}_i) + \\ &\quad + \frac{(\lambda - \lambda_k)}{\lambda} \left[-\tau \sum_{i \in \tilde{\mathcal{R}}} (h_k(\tilde{\mathbf{x}}_i) - \hat{f}(\lambda_k)(\tilde{\mathbf{x}}_i)) + (1-\tau) \sum_{i \in \tilde{\mathcal{L}}} (h_k(\tilde{\mathbf{x}}_i) - \hat{f}(\lambda_k)(\tilde{\mathbf{x}}_i)) \right] \end{aligned}$$

where k is such that $\lambda_{k-1} \leq \lambda \leq \lambda_k$ (where the list of $\lambda_\cdot$'s now combines both knots and validation crossings), and we take advantage of the representation in (6). From the last two rows we can see that L_{cv} is monotone in λ as long as $\tilde{\mathcal{L}}, \tilde{\mathcal{R}}$ are fixed (i.e., no validation crossing occurs) and h_k is fixed (i.e., no knot is encountered). Therefore any local (or global) extremum must be at a knot or a validation crossing. ■

Corollary 4 *Given the complete solution path for $\tau = \tau_0$, the solutions of the bi-level Problem (4) for a range of quantiles around τ_0 can be obtained by following the paths of the knots and the validation crossings only, as τ changes.*

To implement this corollary in practice, we have two main issues to resolve:

1. How do we follow the paths of the validation crossings?
2. How do we determine which one of the knots and validation crossings is going to be optimal for every value of τ?

The first question is easy to answer when we consider the similarity between the knot following problem we solve in Theorem 2 and the validation crossing following problem. In each case we

3. In pathological cases there may be a "segment" of minimizers. In this case it can be shown that such a segment will always be flanked by points described in the theorem.

have a set of *elbow* observations whose fit must remain fixed as τ changes, but whose $\hat{\theta}$ values may vary; sets $\mathcal{L}, \mathcal{R}$ whose $\hat{\theta}$ are changing in a pre-determined manner with τ, but whose fit may vary freely; and one special observation which *characterizes* the knot or validation crossing. The only difference is that in a knot this is a *border* observation from the training set, so both its fit and its $\hat{\theta}$ are pre-determined, while in the case of validation crossing it is a *validation* observation, whose fit must remain fixed (at the *elbow*), but which does not even have a $\hat{\theta}$ value. Taking all of this into account, it is easy to show the following result, closely related to Theorem 2. Assume there is a validation crossing at λ_v for quantile τ_0, and that validation set observation j_v is the one crossing the elbow, that is,

$$\hat{f}(\tau_0, \lambda_v)(\tilde{\mathbf{x}}_{j_v}) = 0 \, .$$

Let $\mathbf{s}_{\mathcal{E}_v}, K_{\mathcal{E}_v}, \mathbf{b}_{\mathcal{E}_v}$ be defined as in Theorem 2 (with $\{i_v\} = \Phi$ for definition of $\mathbf{s}$). Let $\mathbf{k}_v = (K(X_{\mathcal{E}_v}, \tilde{\mathbf{x}}_{j_v}))$ be a $1 \times |\mathcal{E}_v|$ vector of the kernel evaluations at $\tilde{\mathbf{x}}_{j_v}$ for the elbow observation functionals.

Proposition 5 *λ_v moves linearly as τ changes. That is, there exists a constant d_v such that for quantile $\tau_0 + \delta$ there is a validation crossing in the λ-solution path at $\lambda_v + d_v\delta$, for $\delta \in [-\varepsilon_v, \nu_v]$, a non-empty neighborhood of 0. d_v is determined through the solution of a set of $|\mathcal{E}_v| + 2$ linear equations with $|\mathcal{E}_v| + 2$ unknowns:*

$$\mathbf{B}^v \begin{pmatrix} \tilde{\mathbf{b}}^v \\ d_v \end{pmatrix} = \begin{pmatrix} -(|\mathcal{R}| + |\mathcal{L}|) \\ -\mathbf{s}_{\mathcal{E}_v} \\ -\tilde{s}_{j_v} \end{pmatrix}$$

with

$$\mathbf{B}^v = \begin{pmatrix} 0 & \mathbf{1}^\intercal & 0 \\ \mathbf{1} & \mathbf{K}_{\mathcal{E}_v} & -\mathbf{y}_{\mathcal{E}_v} \\ 1 & \mathbf{k}_v & -\tilde{y}_{j_v} \end{pmatrix} \, .$$

Furthermore, the solution $\hat{f}(\tau_0 + \delta, \lambda_v + c_v\delta)$ is given by:

$$\hat{f}(\tau_0 + \delta, \lambda_v + c_v\delta) = \frac{1}{\lambda_v + c_v\delta} \left(\lambda_v \hat{f}(\lambda_v, \tau_0)(\mathbf{x}) + \delta h_v(\mathbf{x}) \right)$$
$$h_v(\mathbf{x}) = \tilde{b}_0^v + \sum_{i \in \mathcal{E}_v} b_i^v K(\mathbf{x}, \mathbf{x}_i) + \sum_{i \in \mathcal{L} \cup \mathcal{R}} K(\mathbf{x}, \mathbf{x}_i) \, .$$

The proof relies on following the same steps as the proof of Theorem 2 and is omitted for brevity.

The second question we have posed requires us to explicitly express the validation loss (i.e., L_τ on the validation set) at every knot and validation crossing in terms of δ, so we can compare them and identify the optimum at every value of δ. Using the representation in (7) we can write the *validation loss* for a knot k (denote $f^k(\delta) = \hat{f}(\tau_0 + \delta, \lambda_k + c_k\delta)$):

$$\begin{aligned} \textstyle\sum_{i=1}^N \quad & L_{\text{cv}}(\tilde{y}_i, f^k(\delta)(\tilde{\mathbf{x}}_i)) = \\ & = -(1 - \tau_0 - \delta) \sum_{i \in \mathcal{L}} (\tilde{y}_i - f^k(\delta)(\tilde{\mathbf{x}}_i)) + (\tau_0 + \delta) \sum_{i \in \mathcal{R}} (\tilde{y}_i - f^k(\delta)(\tilde{\mathbf{x}}_i)) = \\ & = \sum_{i=1}^N L_{\text{cv}}(\tilde{y}_i, f^k(0)(\tilde{\mathbf{x}}_i)) + \delta \sum_i |\tilde{y}_i - f^k(0)(\tilde{\mathbf{x}}_i)| + \frac{\delta}{\lambda_k + c_k\delta} \cdot \qquad (10) \\ & \cdot \left[-(1 - \tau_0 - \delta) \sum_{i \in \mathcal{L}} (c_k f^k(0)(\tilde{\mathbf{x}}_i) - h_k(\tilde{\mathbf{x}}_i)) + (\tau_0 + \delta) \sum_{i \in \mathcal{R}} (c_k f^k(0)(\tilde{\mathbf{x}}_i) - h_k(\tilde{\mathbf{x}}_i)) \right] \, . \end{aligned}$$

A similar expression can be derived for validation crossings (with f^k, c_k, h_k replaced by f^v, d_v, h_v in the obvious way). These are rational functions of δ with quadratic expressions in the numerator and linear expressions in the denominator. Our cross-validation task can be re-formulated as the identification of the minimum of these rational functions among all knots and validation crossings, for every value of τ in the current *segment*, where the directions h_k, h_v of all knots and validation crossings are fixed (and therefore so are the coefficients in the rational functions). This is a *lower-envelope* tracking problem, which has been extensively studied in the literature (Sharir and Agarwal 1995 and references therein). The algorithms developed mostly conform to the common-sense approach of maintaining the order of the validation loss scores from smallest to largest; identifying the τ values where elements with neighboring scores *meet* (i.e., obtain identical score); and whenever a meeting occurs, re-calculating only the relevant crossing points, that is, those of the elements that changed order and their immediate neighbors. We also have to re-calculate some of the validation loss scores whenever an *event* happens on the training solution path (like a *knot crossing*).

To calculate the meeting point of two elements with neighboring scores (assume WLOG that they are two knots k, l) we find the zeros of the cubic equation obtained by requiring equality for the two rational functions of the form (10) corresponding to the two elements. Writing the expression in (10) for both k and l, and requiring equality gives us the cubic equation:

$$\begin{aligned} 0 = \quad & \lambda_k\lambda_l(lo_k - lo_l) + \\ & +\delta[(\lambda_k c_l + \lambda_l c_k)(lo_k - lo_l) + \lambda_k\lambda_l(la_k - la_l) + \lambda_l(\tau_0 LR_k - Le_k) - \lambda_k(\tau_0 LR_l - Le_l)] \\ & +\delta^2[c_k c_l(lo_k - lo_l) + (c_k\lambda_l + c_l\lambda_k)(la_k - la_l) + \\ & \quad +c_l(\tau_0 LR_k - Le_k) + \lambda_l LR_k - c_k(\tau_0 LR_l - Le_l) - \lambda_k LR_l] \\ & +\delta^3[c_k c_l(la_k - la_l) + c_l LR_k - c_k LR_l] \,, \end{aligned} \tag{11}$$

where:

$$\begin{aligned} lo_k &= \sum_{i=1}^{N} L_{\mathrm{cv}}(\tilde{y}_i, f^k(0)(\tilde{\mathbf{x}}_i)) \\ la_k &= \sum_{i} |\tilde{y}_i - f^k(0)(\tilde{\mathbf{x}}_i)| \\ LR_k &= \sum_{i\in\mathcal{L}} (c_k f^k(0)(\tilde{\mathbf{x}}_i) - h_k(\tilde{\mathbf{x}}_i)) + \sum_{i\in\mathcal{R}} (c_k f^k(0)(\tilde{\mathbf{x}}_i) - h_k(\tilde{\mathbf{x}}_i)) \\ Le_k &= \sum_{i\in\mathcal{L}} (c_k f^k(0)(\tilde{\mathbf{x}}_i) - h_k(\tilde{\mathbf{x}}_i)) \,, \end{aligned}$$

with similar expressions for the elements with subscript l derived in the obvious way. The smallest non-negative solution for δ is the one we are interested in.

Figure 3 gives a simple illustration of the process of following the validation loss scores, and identifying their optimum, while updating the directions of the knots and validation crossings as events occur. It shows the set of training and validation loss scores for two knots and the validation loss only for one validation crossing. The training loss is shown in solid lines, and the validation loss in dashed lines. Assuming these are the only three candidates in Theorem 3, the figure shows in bold the lower envelope which defines the optimal cross validation solution at every value of τ. As we can see, in this example the first (left) switch is between two knots as a result of a knot crossing, while the second (right) is a result of a validation crossing becoming optimal. It should be noted,

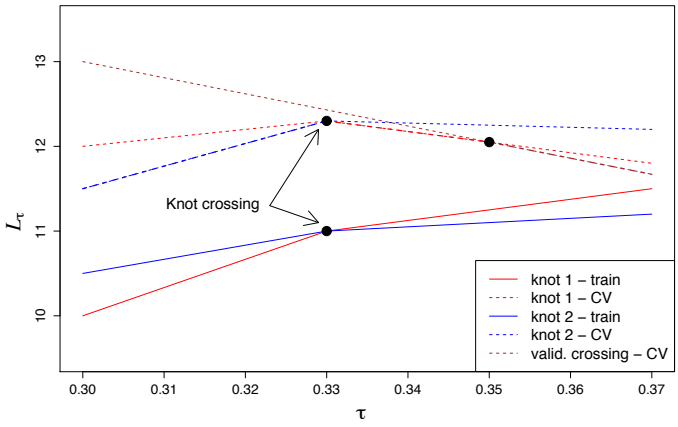


Figure 3: Illustration of the process of lower envelope tracking, in the presence of two knots and one validation crossing being tracked. See text for details.

that the linearity of the solid and dashed lines in Figure 3 is for illustration simplicity, and is not a realistic depiction of the non-linear evolution of the loss as τ varies, as discussed above.

4. Algorithm Overview

Bringing together all the elements from the previous sections, we now give a succinct overview of the resulting algorithm (Algorithm 1). Since there is a multitude of details, we defer a detailed pseudo-code description of our algorithm to Appendix A.

The algorithm follows the knots of the λ-solution path as τ changes using the results of Section 2, and keeps track of the cross-validated solution using the results of Section 3. Every time an *event* happens (like a knot crossing), the direction in which two of the knots are moving has to be changed, or knots have to be added or deleted. Between these events, the evolution of the cross-validation objective at all knots and validation crossings has to be sorted and followed. Their order is maintained and updated whenever crossings occur between them.

4.1 Approximate Computational Complexity

Looking at Algorithm 1, we should consider the number of steps of the two loops and the complexity of the operations inside the loops. Even for a "standard" λ-path following problem for fixed τ, it is in fact impossible to rigorously bound the number of steps in the general case, but it has been argued and empirically demonstrated by several authors that the number of knots in the path behaves as $O(n)$, the number of samples (Rosset and Zhu, 2007; Hastie et al., 2004; Li et al., 2007). In our case the outer loop of Algorithm 1 implements a 2-dimensional path following problem, that can be thought of as following $O(n)$ 1-dimensional paths traversed by the knots of the path. It therefore stands to reason (and we confirm it empirically below) that the outer loop typically has $O(n^2)$ steps where *events* happen. The events in the inner loop, in turn, have to do with the N validation observations meeting the $O(n)$ knots. So a similar logic would lead us to assume that the number of meeting events (counted by the inner loop) should be at most $O(nN)$ total for the whole running

Algorithm 1: Main steps of our algorithm

Input: The entire λ-solution path for quantile τ_0; the bi-level optimizer $\lambda^*(\tau_0)$
Output: Cross-validated solutions $f^*(\tau)$ for $\tau \in [\tau_0, \tau_{\text{end}}]$

Initialization: Identify all knots and validation crossings in the solution path for τ_0; Find direction of each knot according to Theorem 2 ;
Find direction of each validation crossing according to Proposition 5;
Create a list M of knots and validation crossings sorted by their validation loss ;
Let $\lambda^*(\tau_0)$ be the one at the bottom of the list M, and $f^*(\tau_0)$ accordingly ;
Calculate future meeting of each pair of neighbors in M by solving the cubic equation implied by (10);
Set $\tau_{\text{now}} = \tau_0$;
while $\tau_{\text{now}} < \tau_{\text{end}}$ **do**
    Find value $\tau_1 > \tau_{\text{now}}$ where first knot crossing occurs;
    Find value $\tau_2 > \tau_{\text{now}}$ where first knot merge occurs;
    Find value $\tau_3 > \tau_{\text{now}}$ where first knot birth occurs;
    Set $\tau_{\text{new}} = \min(\tau_1, \tau_2, \tau_3)$;
    while $\tau_{\text{now}} < \tau_{\text{new}}$ **do**
        Find value $\tau_4 > \tau_{\text{now}}$ where first future meeting (order change) in M occurs;
        Find value $\tau_5 > \tau_{\text{now}}$ where first validation crossing birth occurs;
        Find value $\tau_6 > \tau_{\text{now}}$ where first validation crossing cancelation occurs;
        Set $\tau_{\text{next}} = \min(\tau_4, \tau_5, \tau_6, \tau_{\text{new}})$;
        Update $\lambda^*(\tau), f^*(\tau)$ for $\tau \in (\tau_{\text{now}}, \tau_{\text{next}})$ as the evolution of the knot or validation crossing attaining the minimal L_{cv} in M (i.e., the one at $\lambda^*(\tau_{\text{now}})$) ;
        Update M according to the first event (order change, birth, cancelation);
        Update the future meetings of the affected elements using (10);
        Set $\tau_{\text{now}} = \tau_{\text{next}}$;
    end
    Update the list of knots according to the first event (knot crossing, birth, merge) ;
    Update the directions of affected knots using Theorem 2 ;
end

of the algorithm (i.e., many iterations of the outer loop may have no events happening in the inner loop). Each iteration of either loop requires a re-calculation of up to three directions (of knots or validation crossings), using Theorem 2 or Proposition 5. These calculations involve updating and inverting matrices that are roughly $|\mathcal{E}| \times |\mathcal{E}|$ in size (where $|\mathcal{E}|$ is the number of observations in the elbow). However note that only one row and column are involved in the updating, leading to a complexity of $O(n + |\mathcal{E}|^2)$ for the whole direction calculation operation, using the Sherman-Morrison formula (Sherman and Morrison, 1949) for updating the inverse. In principle, $|\mathcal{E}|$ can be equal to n, although it is typically much smaller for most of the steps of the algorithm, on the order of $\sqrt{n}$ or less. So we assume here that the loop cost is between $O(n)$ and $O(n^2)$.

Putting all of these facts and assumptions together, we can estimate the algorithm's computational complexity's *typical* dependence on the number of observations in the training and validation set as ranging between $O(n^2 \cdot \max(n, N))$ and $O(n^3 \cdot \max(n, N))$. Clearly, this estimation procedure

falls well short of a formal "worst case" complexity calculation, but we offer it as an intuitive guide to support our experiments below and get an idea of the dependence of running time on the amount of data used.

We have not considered the complexity of the lower envelope tracking problem in our analysis, because it is expected to have a much lower complexity (number of order changes $O(\max(n,N)\log(\max(n,N)))$ and each order change involves $O(1)$ work).

5. Extensions

In this section we discuss some of the possible extensions of our algorithm. First we discuss the design of algorithms that are similar in spirit for other parameterized loss function problems, in particular for support vector regression (ε-SVR) and *Huberized* least squares regression. We then move on to the use of in-sample model selection criteria instead of cross validation. Finally, we address the issue of possible non-monotonicity in τ, noted by previous authors (Koenker, 2005; Takeuchi et al., 2006). We demonstrate how our algorithm can be naturally extended to amend this situation.

5.1 Support Vector Regression and Weighted Support Vector Machines

One possible view of regularized ε-SVR (Smola and Schölkopf, 2004) is similar to the quantile regression problem for $\tau = 0.5$:

$$\hat{f}(\varepsilon,\lambda) = \arg\min_f \sum_i L_\varepsilon(y_i - f(\mathbf{x}_i)) + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}_K} \tag{12}$$

where the parameterized loss function L_ε is piecewise linear and symmetric around zero, with a *don't care* region of size 2ε:

$$L_\varepsilon(r) = \begin{cases} r-\varepsilon & r \geq \varepsilon \\ 0 & -\varepsilon < r < \varepsilon \\ -r-\varepsilon & r \leq -\varepsilon \end{cases} .$$

The loss is parameterized by ε. From an optimization perspective, this problem is very similar to KQR, with a piecewise linear loss and an RKHS norm penalty. The solution can be represented as in (5), and the KKT conditions for optimality of solutions of (12) in terms of coefficients of representer functions have been formalized and used to design λ-path following algorithms by Gunther and Zhu (2005). For example, if we define for a proposed solution $f(\mathbf{x})$ of ε-SVR:

- $\mathcal{L} = \{i : y_i - f(\mathbf{x}_i) < -\varepsilon\}$ (points on left of *left elbow* of L_ε)
- $\mathcal{E}_\mathcal{L} = \{i : y_i - f(\mathbf{x}_i) = -\varepsilon\}$ (*left elbow*)
- $\mathcal{C} = \{i : |y_i - f(\mathbf{x}_i)| < \varepsilon\}$ (don't care region)
- $\mathcal{E}_\mathcal{R} = \{i : y_i - f(\mathbf{x}_i) = \varepsilon\}$ (*right elbow*)
- $\mathcal{R} = \{i : y_i - f(\mathbf{x}_i) > 0\}$ (right of *right elbow*),

Then the Karush-Kuhn-Tucker (KKT) conditions for optimality of a solution $\hat{f}(\varepsilon,\lambda)$ of Problem (12) can be phrased as:

- $i \in \mathcal{C} \;\Rightarrow\; \hat{\theta}_i = 0$
- $i \in \mathcal{L} \;\Rightarrow\; \hat{\theta}_i = -1$
- $i \in \mathcal{R} \;\Rightarrow\; \hat{\theta}_i = 1$
- $i \in \mathcal{E}_{\mathcal{L}} \;\Rightarrow\; -1 \leq \hat{\theta}_i \leq 0$
- $i \in \mathcal{E}_{\mathcal{R}} \;\Rightarrow\; 0 \leq \hat{\theta}_i \leq 1$
- $\sum_i \hat{\theta}_i = 0$.

Wang et al. (2006) have noted that ε-paths (with fixed λ) can similarly be followed. Because of the fundamental similarity in the optimization setting, all our results regarding behavior of λ-paths and knots as τ changes in quantile regression (e.g., Theorem 2) can be adapted in a reasonably straight forward manner to follow paths and knots of λ-solution paths in ε-SVR, as ε varies.

There is, however, a fundamental difference in the statistical setting between parameterized quantile loss and parameterized ε-SVR loss. While every quantile loss function defines an interesting modeling problem of estimation of a given conditional quantile, there is no such clear motivation for varying ε. Furthermore, there is no obvious way in which the cross validation loss should change with ε, if at all. In most cases, it seems ε is viewed more as another tuning parameter for a single modeling problem (like λ), than a parameter defining a range of loss functions, each of its own independent interest. In this situation, the only motivation for solving the range of bi-level problems parameterized by ε may be as a way to efficiently traverse the entire (ε, λ) solution space in search of a single "best" prediction model. It may therefore be appropriate to use a single cross validation objective L_{cv} independent of ε. If we choose $L_{\text{cv}} = L_{\tau=0.5}$ (the symmetric quantile loss, sometimes called absolute loss), then our observations on the bi-level path following problem (e.g., Theorem 3) would require slight modifications, but the ideas would carry through to the ε-SVR case in a straight forward manner.

An interesting recent development is the proposal of weighted support vector machines for probability estimation by Wang et al. (2008). Their proposed approach calls for fitting weighted versions of the support vector machine, with a range of relative weights applied to the two classes, as a provably valuable approach for estimating probabilities. We omit the details for brevity, but note that like the SVR case above, extending our bi-level approach to this problem is straight forward.

5.2 ℓ_1-regularized Huberized Squared Loss

Rosset and Zhu (2007) suggested the use of robust versions of squared error loss with ℓ_1 regularization, as an approach for combining computational efficiency and robustness to long-tailed error distribution. Huber's loss function is quadratic for small absolute residuals, then continues linearly as the residuals move away from zero, while maintaining differentiability. It is parameterized with a *huberizing point* t:

$$L_t(r) = \begin{cases} r^2 & |r| < t \\ 2t|r| - r^2 & \text{otherwise} \end{cases}.$$

The algorithm proposed in Rosset and Zhu (2007) for λ-path following can be thought of as an extension of the LARS-Lasso algorithm proposed for the Lasso (squared error loss with ℓ_1 penalty) by Efron et al. (2004). The loss function is differentiable, there is no concept of *elbow* (although

there are still knots), the KKT conditions are quite different, and if we also use a differentiable L_{cv}, the cross validation procedure would be affect as well. However, the general reasoning of this paper can still be applied to build bi-level path following algorithms for the Huberized lasso problem, and to choose good t, λ combinations.

5.3 Use of In-sample Model Selection Criteria

Li et al. (2007) follow the literature in proposing two model selection criteria for selecting λ^* for a fixed value of τ, when there is no validation sample. These are Schwartz information criterion (SIC, Schwarz, 1978) and generalized approximate cross validation (GACV, Yuan, 2006). Both of these use the model's effective degrees of freedom (DF) as a complexity measure which penalizes the empirical error. Following Zou et al. (2007), Li et al. (2007) show that an unbiased estimate of DF is the size of the elbow $|\mathcal{E}|$. Thus, they arrive at following SIC and GACV approximations:

$$\text{SIC}(\lambda) = \log\left(\frac{1}{n}\sum_{i=1}^{n} L_\tau(y_i - \hat{f}(\tau,\lambda)(\mathbf{x}_i))\right) + \frac{\log n}{2n}|\mathcal{E}| \quad (13)$$

$$\text{GACV}(\lambda) = \frac{\sum_{i=1}^{n} L_\tau(y_i - \hat{f}(\tau,\lambda)(\mathbf{x}_i))}{n - |\mathcal{E}|}\,. \quad (14)$$

If we were to adopt these measures (or similar ones) for model selection instead of the cross validation approach using an independent validation set, tracking the optimal solution $\lambda^*(\tau)$ requires no extra work besides following the knots of the solution (as described in Sections 2, 4). This is guaranteed by the following simple result:

Proposition 6 *For any fixed τ, the minimizer $\lambda^*(\tau)$ of SIC, GACV and any similar model selection criterion which is monotone in both $\sum_{i=1}^{n} L_\tau(y_i - \hat{f}(\tau,\lambda)(\mathbf{x}_i))$ and $|\mathcal{E}|$, is always at one of the knots of the solution path.*

Proof The loss is monotone between knots (e.g., from looking at Equation 6), while $|\mathcal{E}|$ is fixed. ■

Thus, if we wish to use SIC or similar measures for selecting $\lambda^*(\tau)$, the inner loop of Algorithm 1 (lines 12-21) can be omitted and replaced with a simple tracking of the value of SIC at the knots that are being followed. Since the algorithmic complexity of applying SIC/GACV is reduced compared to cross validation, and given the ongoing debates in the literature on the merits of in-sample versus out-of-sample model selection, it may often be beneficial to apply these in-sample methods in addition, or even instead of, cross validation. We demonstrate and compare performance of the two approaches in Section 6 below.

5.4 Addressing *Quantile Crossings*

The problem of quantile crossing, as formulated by Koenker (2005), is that for any fixed λ (in particular $\lambda = 0$ in the linear quantile regression case, which is the one that Koenker (2005) concentrates on), the prediction $\hat{f}(\tau,\lambda)(\mathbf{x})$ may not be non-decreasing in τ for a fixed $\mathbf{x}$. That is, we may have $\tau_0 < \tau_1$ and $\hat{f}(\tau_0,\lambda)(\mathbf{x}) > \hat{f}(\tau_1,\lambda)(\mathbf{x})$, which can never be true of the corresponding population conditional quantiles, of course.

Takeuchi et al. (2006) address this problem by constraining the solution to comply with the monotonicity requirement over a *finite* set of "interesting" quantiles. Their approach cannot work

in our case, since our algorithm is local in nature and generates the solutions for the complete space of (τ,λ) values. However, we can offer a partial remedy to the quantile crossing problem through observation of the guaranteed sub-optimality of the resulting solutions, and a consequent *envelope tracking* modification. The main motivation is the following:

Proposition 7 *Assume* $\tau_0 < \tau_1$ *and* $\hat{f}(\tau_0,\lambda)(\mathbf{x}) > \hat{f}(\tau_1,\lambda)(\mathbf{x})$ *for some* $\lambda,\mathbf{x}$. *Then either*

$$\mathbb{E}_{Y|X=\mathbf{x}} L_{\tau_0}(Y,\hat{f}(\tau_0,\lambda)(\mathbf{x})) \geq \mathbb{E}_{Y|X=\mathbf{x}} L_{\tau_0}(Y,\hat{f}(\tau_1,\lambda)(\mathbf{x}))\,. \tag{15}$$

or

$$\mathbb{E}_{Y|X=\mathbf{x}} L_{\tau_1}(Y,\hat{f}(\tau_0,\lambda)(\mathbf{x})) \leq \mathbb{E}_{Y|X=\mathbf{x}} L_{\tau_1}(Y,\hat{f}(\tau_1,\lambda)(\mathbf{x})) \tag{16}$$

Thus, we can always improve the predictive quality of either $\hat{f}(\tau_0,\lambda)$ *or* $\hat{f}(\tau_1,\lambda)$ *by eliminating the non-monotonicity.*

Proof In what follows we eliminate the explicit conditioning in the expectations. All expectations are with regard to the distribution $P(Y|X=\mathbf{x})$. Denote by c_0 and c_1 the τ_0 and τ_1 quantiles respectively of $P(Y|X=\mathbf{x})$. By definition, $c_0 \leq c_1$. We also assume $\hat{f}(\tau_0,\lambda)(\mathbf{x}) > \hat{f}(\tau_1,\lambda)(\mathbf{x})$. We hereafter denote these two fitted value by $\hat{f}_0,\hat{f}_1$ respectively for brevity. This gives us three possible scenarios:

1. $\hat{f}_1 \geq c_0$. In this case (15) holds, since:

$$\begin{aligned}
\mathbb{E}L_{\tau_0}(Y,\hat{f}_0) &= \tau_0 \int_{y\geq\hat{f}_0} y-\hat{f}_0 dP(y|\mathbf{x}) + (1-\tau_0)\int_{y<\hat{f}_0} -y+\hat{f}_0 dP(y|\mathbf{x}) \\
&= \tau_0 \int_{y\geq\hat{f}_0} P(Y\geq y|\mathbf{x})dy + (1-\tau_0)\int_{y<\hat{f}_0} P(Y\leq y|\mathbf{x})dy \\
&= \mathbb{E}L_{\tau_0}(Y,\hat{f}_1) + \int_{\hat{f}_1}^{\hat{f}_0} \left[(1-\tau_0)P(Y\leq y|\mathbf{x}) - \tau_0 P(Y\geq y|\mathbf{x})\right] dy \\
&\geq \mathbb{E}L_{\tau_0}(Y,\hat{f}_1)\,,
\end{aligned}$$

   where the inequality on the last line is because $P(Y\leq y|X=\mathbf{x}) \geq \tau_0$ in the range $\hat{f}_1 \leq y \leq \hat{f}_0$ (by our assumption that $c_0 \leq \hat{f}_1 < \hat{f}_0$).

2. $\hat{f}_0 \leq c1$. By the same line of argument in this case (16) holds.

3. If neither of the previous two holds, we must have $\hat{f}_1 < c_0 \leq c_1 < \hat{f}_0$. Following the same steps as in case 1 we write:

$$\mathbb{E}L_{\tau_0}(Y,\hat{f}_0) = \mathbb{E}L_{\tau_0}(Y,\hat{f}_1) + \int_{\hat{f}_1}^{\hat{f}_0} \left[(1-\tau_0)P(Y\leq y|\mathbf{x}) - \tau_0 P(Y\geq y|\mathbf{x})\right] dy \tag{17}$$

$$\mathbb{E}L_{\tau_1}(Y,\hat{f}_0) = \mathbb{E}L_{\tau_1}(Y,\hat{f}_1) + \int_{\hat{f}_1}^{\hat{f}_0} \left[(1-\tau_1)P(Y\leq y|\mathbf{x}) - \tau_1 P(Y\geq y|\mathbf{x})\right] dy\,. \tag{18}$$

   Assume $\mathbb{E}L_{\tau_0}(Y,\hat{f}_0) < \mathbb{E}L_{\tau_0}(Y,\hat{f}_1)$. It implies the integral in (17) is negative which in turn implies that the integral in (18) is also negative, since trivially

$$\frac{\partial}{\partial\tau}\int_{\hat{f}_1}^{\hat{f}_0} \left[(1-\tau)P(Y\leq y|\mathbf{x}) - \tau P(Y\geq y|\mathbf{x})\right] dy < 0\,.$$

   This negativity implies $\mathbb{E}L_{\tau_1}(Y,\hat{f}_0) < EL_{\tau_1}(Y,\hat{f}_1)$.

■

The following is an immediate consequence of Proposition 7 if we take $P(Y|\tilde{\mathbf{x}}_i)$ to be a point mass at $Y = \tilde{y}_i$.

Corollary 8 *If non-monotonicity holds at a validation point, that is,* $\tau_0 < \tau_1$ *and* $\hat{f}(\tau_0,\lambda)(\tilde{\mathbf{x}}_i) > \hat{f}(\tau_1,\lambda)(\tilde{\mathbf{x}}_i)$*, then either*

$$L_{\tau_0}(\tilde{y}_i, \hat{f}(\tau_0,\lambda)(\tilde{\mathbf{x}}_i)) \geq L_{\tau_0}(\tilde{y}_i, \hat{f}(\tau_1,\lambda)(\tilde{\mathbf{x}}_i))$$

or

$$L_{\tau_1}(\tilde{y}_i, \hat{f}(\tau_0,\lambda)(\tilde{\mathbf{x}}_i)) \leq L_{\tau_1}(\tilde{y}_i, \hat{f}(\tau_1,\lambda)(\tilde{\mathbf{x}}_i)) \ .$$

Thus, we can improve our holdout performance at either quantile τ_0 *or* τ_1 *by appropriately enforcing monotonicity.*

We conclude that eliminating non-monotonicity can improve both predictive performance and cross validation performance. In terms of practical implications, it is easy to see (though not trivial to implement) how our algorithm can be extended to identify quantile crossings. When these occur, at least one knot will be moving in the 'wrong direction', that is, the expression in (7) will be decreasing in δ. The algorithm will then have to keep careful tabs on the upper and lower limits of the fit at every λ as τ changes (the quantile-crossing gap). Discussion of the details and the appropriate way to resolve the non-monotonicity given this envelope is left for future work.

6. Experiments

Our methodology offers a new approach for generating the full set of cross-validated kernel quantile regression models. There are several interesting aspects of the modeling problem in general and our algorithm in particular that should be studied through a data-based study.

First, to evaluate the new algorithm, the efficiency of the algorithm should be compared to alternative approaches that allow generation of complete set of solutions and cross-validation. This includes the naive *grid-based* search whereby the KQR problem is solved using standard approaches (Takeuchi et al., 2006) for a grid of values in the (τ,λ) space, and a good regularization parameter is chosen for each value of τ by cross-validation; and the method of Li et al. (2007), which can be used to generate the complete λ-path at a grid of τ-values and cross validate each path separately. As Li et al. (2007) demonstrated clearly, their λ-path method is far superior to the grid-based approach in terms of computation, and so we concentrate on comparison to the λ-path approach only, and show that our algorithm compares favorably to it in generating the full set of bi-level solutions.

Second, we may also be interested in studying properties of the modeling problem, not necessarily tied to the new algorithm. Cross-validation based selection of regularization should be compared to *in-sample* approaches such as SIC (Schwarz, 1978) and GACV (Yuan, 2006). As noted above, all of these can be implemented in our framework. It is obvious that given the same amount of data for model fitting, it is better to use holdout data for model selection. However, the fair comparison should be between integrating the validation set into the training set and implementing an in-sample model selection approach, and using a smaller training set in a cross-validation framework.

Another interesting question about the modeling approach regards the ability of KQR to deal with skewed and non-homogeneous error distributions, and still generate reasonable estimates of the underlying quantiles.

We address all of these aspects in this section.

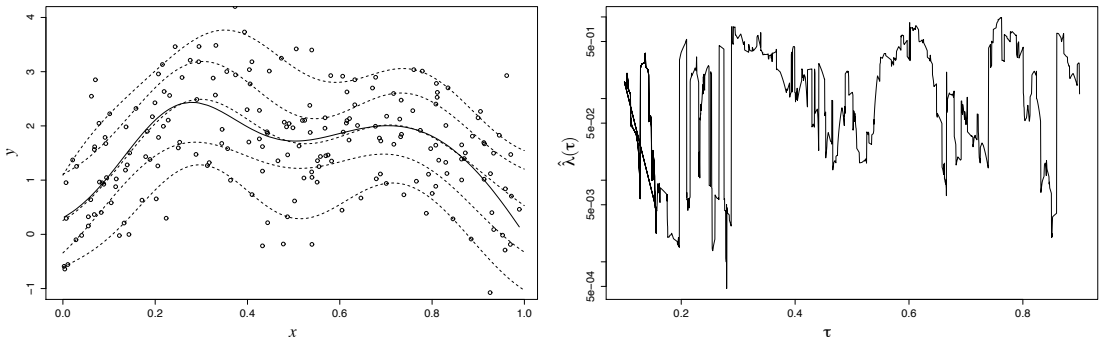


Figure 4: Left: The function $f(x)$ (solid), data points drawn from it with i.i.d normal error, and our cross-validated estimates of quantiles $0.1, 0.25, 0.5, 0.75, 0.9$ (dashed lines, from bottom to top). Right: Evolution of optimal regularization parameter $\hat{\lambda}(\tau)$, as τ varies.

6.1 Simulations

Our simulation setup starts with univariate data $x \in [0,1]$ and a "generating" function $f(x) = 2 \cdot \left(\exp(-30 \cdot (x-0.25)^2) + \sin(\pi \cdot x^2)\right)$ (see Figure 4). We then let $Y = f(x) + \varepsilon$, where the errors ε are independent, with a distribution that can be either:

1. $\varepsilon \sim N(0,1)$, that is, i.i.d standard normal errors
2. $\varepsilon + (x+1)^2 \sim \exp(1/(x+1)^2)$, which gives us errors that are still independent and have mean 0, but are asymmetric and have non-constant variance, with small signal-to-noise ratio on the higher values of x (see Figure 5).

Figure 4 demonstrates the results of the algorithm with i.i.d normal errors, 200 training samples and 200 validation samples and a Gaussian kernel with parameter $\sigma = 0.2$. In the left panel, we see that the quantile estimates all capture the general shape of the true curve, with some "smoothing" due to regularization. In the right panel we see the evolution of the optimal regularization parameter $\hat{\lambda}(\tau)$ as τ varies. We see the expected "jumpy" behavior of the optimal parameter, but we do not see a clear tendency to be smaller for quantiles closer to $1/2$. This is somewhat surprising when we think in terms of bias and variance (or approximation error and estimation error) in learning. Values of τ closer to $1/2$ typically create learning problems that are "easier", that is, variance is smaller (Koenker, 2005), and this should in principle allow us to build more complex models (reduce regularization), and decrease bias. A confounding factor in this analysis is the fact that the scale of quantile error need not be comparable for different quantiles. In particular, we may expect that loss magnitude would be larger for quantiles close to 0.5, where both types of errors get penalized equally. If that is the case, then having the similar regularization parameter may in fact imply *less* regularization for τ close to 0.5 compared to extreme quantiles. Another interesting observation is that while $\lambda^*(\tau)$ may be jumpy, both the empirical and the validation loss may vary smoothly. This smoothness is in fact guaranteed for the validation loss L_{cv}, since it is easily seen that the "jumps" are points where validation loss is equal at two knots or validation crossings.

Next we consider the computational complexity of the algorithm, and its dependence on the number of training samples (with 200 validation samples). We compare it to the KQR algorithm

NTRAIN	NSTEPS	TIME(BI-LEVEL)	TIME(LI ET AL.)	BREAK-EVEN RESOLUTION
200	29238	931 SEC.	2500 SEC.	3000
100	12269	99 SEC.	900 SEC.	900
50	2249	23 SEC.	480 SEC.	400

Table 1: Number of steps and run times of our algorithm and of Li et al. (2007), for the whole path from $\tau = 0.1$ to $\tau = 0.9$, as a function of the number of training observations NTRAIN. These results are based on applying Li et al. (2007) at 8000 different values of τ. The last column shows what resolution would give similar running times to both approaches (see text for details).

of Li et al. (2007), who have already demonstrated that their algorithm is significantly more efficient than grid-based approaches for generating 1-dimensional paths for fixed τ. Table 1 shows the number of steps of the main (outer) loop of Algorithm 1 and the total run time of our algorithm for generating the complete set of cross-validated solutions for $\tau \in [0.1, 0.9]$ as a function of the number of training samples (with validation sample fixed at 200). Also shown is the run time for the algorithm of Li et al. (2007), when we use it on a grid of 8000 evenly spaced τ values in $[0.1, 0.9]$ and find the best cross validated solution by enumerating the candidates as identified in Section 3. Our conjecture that the number of knots in the 2-dimensional path behaves like $O(n^2)$ seems to be consistent with these results, as is the hypothesized overall time complexity dependence of $O(n^3)$. Since 8000 is typically an unnecessarily fine grid for practical applications, we offer in the last column an evaluation of the comparative efficiency of the two methods in terms of the number of distinct τ values that can be fitted with the Li et al. (2007) approach in roughly the same running time as our approach. It is clear from these results that if just a small number of τ values (say, 10) are sufficient to address the complete problem, our approach does not carry a computational benefit.

Next, we demonstrate the ability of KQR to capture the quantiles with "strange" errors from model 2. Figure 5 shows a data sample generated from this model and the $(0.25, 0.5, 0.75)$ quantiles of the conditional distribution $P(Y|X)$ (solid), compared to their cross-validated KQR estimates (dashed), using 500 samples for learning and 200 for validation (more data is needed for learning because of the very large variance at values of x close to 1). As expected, we can see that estimation of the lower quantiles, and at smaller values of x is easier, because the distribution $P(Y|X = x)$ has long right tails everywhere and has much larger variance when x is big.

6.2 Baseball Data and California Housing

As discussed in Perlich et al. (2007), estimating conditional quantiles is often a modeling task that is well grounded in practical applications. In the context of house prices, we can think of estimating a high (but not extreme[4]) conditional quantile as the seller's search for a favorable bargaining position in negotiations. Similarly for salaries, estimating a high conditional quantile can serve as a measure of what an employee can expect to receive optimistically (but still realistically), given his characteristics and performance.We therefore demonstrate KQR on two well studied data sets that correspond to such modeling problems: baseball salaries as a function of a player's home runs and years of experience (He et al., 1998) and the California housing data set (Pace and Barry, 1997),

4. Extreme quantile estimation is also of interest in some contexts, but we do not demonstrate it here due to the inherent statistical difficulty and questionable results, see some discussion in Conclusion section.

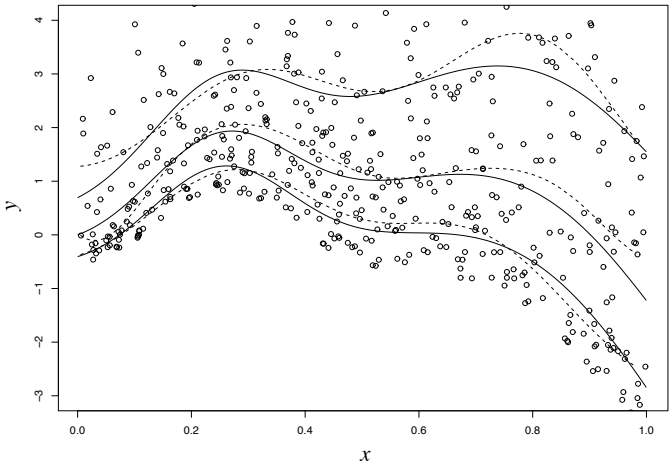


Figure 5: Quantiles of $P(Y|X)$ (solid), and their estimates (dashed) for quantiles $(0.25, 0.5, 0.75)$ with the exponential error model.

which describes the median prices of houses in neighborhoods of California along with nine explanatory demographic variables. We seek to demonstrate predictive performance, fitted models and the relative performance of different model selection approaches.

For our experiments, we use a Gaussian kernel, with the parameter $\gamma = 1$ chosen based on experimentation, to give flexible but not overly jumpy fits. We demonstrate the fit and accuracy of model selection using CV compared to using SIC. For CV, we used 50 of the 263 players in the data set for validation (selection of $\hat{\lambda}(\tau)$) and 50 more for testing the accuracy of the resulting model. Thus, 163 examples were used for training. For SIC, we used 213 (training+validation) as the training set, and applied Equation (13) for selecting $\hat{\lambda}(\tau)$. Both approaches were evaluated using the 50 test observations. In Figure 6 we show the resulting fit in both approaches, for three different quantiles. As expected, compensation seems to be monotone in performance (home runs) but not in experience (salary tends to increase as players gain experience, but then decreases as they get older and performance deteriorates). As we can see, the model-selected surfaces are quite similar between CV and SIC, though this need not be the case, as we should keep in mind that SIC is choosing between models trained on more data. In terms of accuracy on the test set (shown above each plot), The results are also very comparable. When comparing the two approaches we should also keep in mind the reduced complexity of applying SIC, and the existing literature on instability of CV-based model selection, though this is not evident in our results.

For the California housing data set, we use only longitude and latitude as the two explanatory variables in fitting KQR, to facilitate meaningful visualization of results. We model the log of the median price, since the actual median fluctuates widely over the data. We use 500 observations for training and 50 as validation for CV, 550 as training for SIC, and 500 additional observations for testing. Figure 7 shows the results. It is clear that visualization is hampered by the fact that California is far from being rectangular, so one corner of the plots (latitude 34N, longitude 122W) is well inside the ocean, while the other (latitude 40N, longitude 115W) is well inland from the California border. The wild extrapolation of the fit in that direction is therefore not informative.

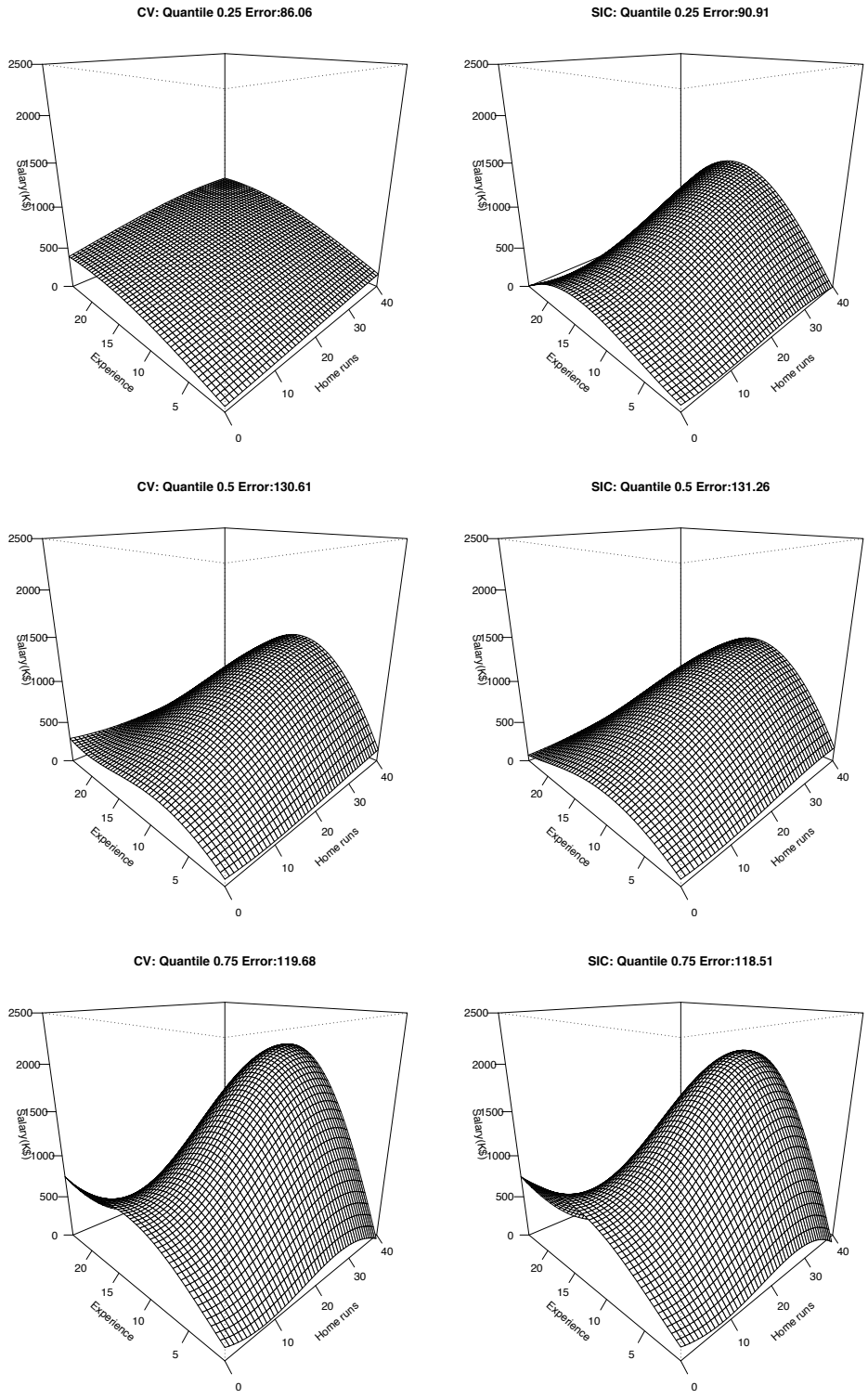


Figure 6: Models selected using CV (left) and SIC (right) on the Baseball data, for three different quantiles.

On each plot the fit at San Francisco (red circle), Los Angeles (blue square) and Sacramento (green triangle) are marked. We can see that the selected fits using CV and SIC are quite similar, with possible exception to the more jumpy fit selected by SIC for quantile 0.5. The valid insights that seem to arise out of these plots relate to the lower house values in the central valley of California compared to the coastal area, and the reduced house values in the Sacramento area compared to near by the Bay Area.

7. Conclusions and Future Work

In this paper we have demonstrated that the family of bi-level optimization Problems (4) defined by the family of loss functions L_τ can be solved via a *path following* approach which essentially maps the whole surface of solutions $\hat{f}(\tau,\lambda)$ as a function of both τ and λ and uses insights about the possible locations of the bi-level optima to efficiently find them. This leads to a closed-form algorithm for finding $f^*(\tau)$ for all quantiles. We see two main contributions in this work: a. Characterization of a family of non-convex optimization problems of great practical interest which can be solved using solely convex optimization techniques and b. Formulation of a practical algorithm for generating the full set of cross-validated solutions for the family of kernel quantile regression problems.

We have shown how our approach can be extended to other modeling problems with a parameterized loss function, such as SVR, and to other versions of KQR, including using in-sample model selection criteria and enforcing monotonicity on the resulting quantiles.

There are many other interesting aspects of our work, which we have not touched on here, including: development of further optimization shortcuts to improve algorithmic efficiency, investigation of the range of applicability of our algorithmic approach beyond KQR and SVR, analysis of the use of various kernels for KQR and how the kernel parameters and kernel properties interact with the solutions, and more extensive empirical studies.

It is of particular interest to us to investigate the bias-variance tradeoff in loss function selection. As we have mentioned, modeling with the quantile loss function L_τ leads to estimation of the τth quantile of $P(Y|x)$ in the *decision theoretic* sense that the population optimizer of the loss function is this quantile (see Equation 2). However, this by no means guarantees that a model learned from finite data using L_τ (with or without regularization) will do well in predicting the τth quantile. In particular, there is no guarantee that a model built using a different loss function (say, L_η, $\eta \neq \tau$) will not do better in predicting this quantile. This can be thought of in terms of bias and variance, where the model generating quantile η is similar enough to the one for quantile τ (i.e., bias is small), but it is "easier" to learn with L_η, that is, variance is smaller, which would typically be the case if η is closer to $1/2$ than τ (Koenker, 2005). A detailed investigation of this question is outside the scope of the current work, but will be a natural extension.

A particularly important and difficult type of quantile estimation problems pertains to estimation of *extreme* quantiles (e.g., $\tau = 0.01$ or $\tau = 0.99$) which can serve as approximations for expected extreme values of the function being estimated. These problems are typically very difficult *statistically*, that is, hard because of the scarcity of information implicit in trying to estimate events we rarely observe. However they are not expected to be particularly difficult algorithmically. That is, our (and others') KQR approaches can estimate these models, but it is not clear how useful the results are. These observations are verified by our limited experiments (results not shown), which yield very "jumpy" and unstable models for extreme quantiles.

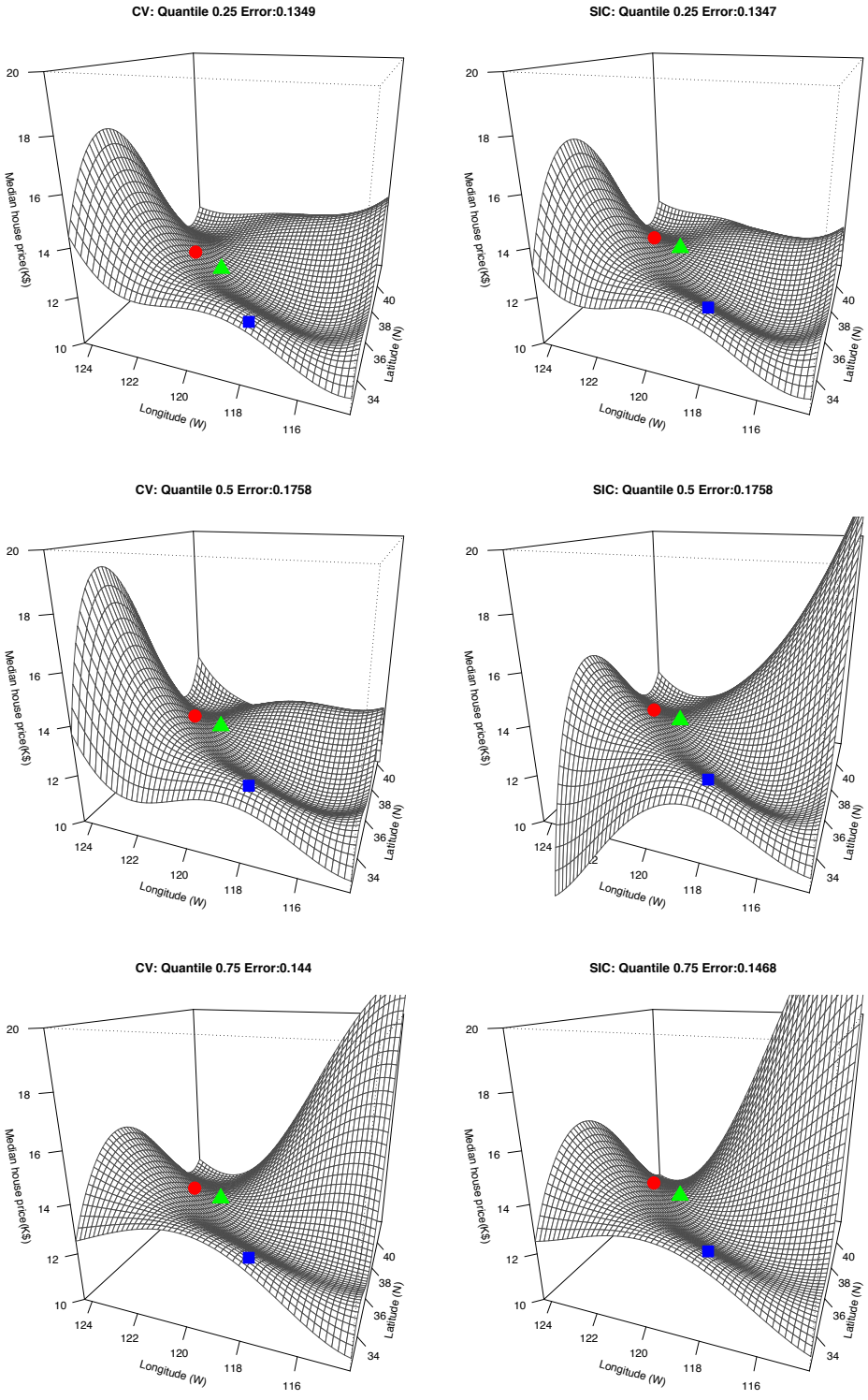


Figure 7: Synthetic maps of the models selected using CV (left) and SIC (right) on the CA housing data, for three different quantiles. The fits at San Francisco (red circle), Los Angeles (blue square) and Sacramento (green triangle) are marked on each map.

Acknowledgments

The author thanks Ramesh Natarajan for useful early discussions, Ji Zhu for help with the KQR code of Li et al. (2007), and the action editor, two anonymous referees and Ronny Luss for their useful comments.

Appendix A. Pseudo-code of Algorithm

Algorithm 2 and its accompanying procedures describe our implementation in some detail. This pseudo code is meant to complete the implementation details given in the paper. We use mathematical notation rather than programming commands as much as possible, to make understanding easier. Given the complexities and intricacies involved in the complete implementation, it seems unrealistic and probably non-useful to give an exhaustive description. Rather, we concentrate on clarifying the general flow of the algorithm and the mathematical problems it solves at each step. We also emphasize the aspects of the algorithm not covered in technical detail in the main text, such as the differentiation of different types of events (knot crossing, knot merging, knot splitting). Where the text offers the technical content, we simply refer to that point. For example, Theorem 2 describes the direction calculation and also implicitly the accounting orf the identities of the sets $\mathcal{E}, \mathcal{L}, \mathcal{R}$ required for it. We thus simply refer back to it where relevant in the algorithm.

Some further comments on the pseudo code:

- We assume the training and validation data are "global variables" known to all procedures.
- We use $\hat{f}$ and $\hat{\theta}$ interchangeably, given formula (5).
- Some of the elements are not described in the most efficient implementation, which would require a lot more accounting and data management. For example, the search for the minima in the function UpdateValidList does not have to be done from scratch on every call, but a list can be maintained, and only the necessary items updated.
- The pseudo-code glosses over numerical issues which plague the actual implementation. In particular, all equalities must have "tolerance" in the practical implementation due to machine rounding errors. This obviously creates a problem when events on the path are bunched together close enough that two distinct events fall within this tolerance.
- We avoid repetition of similar procedures. Thus the call to function *KnotSplit* at end of Algorithm 2 is replaced with a brief explanation of its near-identity to the function *KnotCross* which is already given.

Algorithm 2: Algorithm description

Input: The entire λ-solution path for quantile τ_0 characterized by its m knots $\boldsymbol{\lambda} = \lambda_1, ..., \lambda_m$; the solution directions $\mathbf{g} = g_1, ..., g_m$ as defined in (6); and $\mathbf{i} = i_1, ..., i_m$ the observations which "hit the elbow" at every knot

Output: Cross-validated solutions $f^*(\tau)$ for $\tau \in [\tau_0, \tau_{\text{end}}]$ described as set of intervals in the variable OPT

`/* Initialization: find validation crossings, calculate cross validation loss at knots and validation crossings, sort by it, find future meetings of neighbors on the list, where the order changes */`

Set M = InitializeValidList($\tau_0, \boldsymbol{\lambda}, \mathbf{g}$);

Set $OPT = (\tau_0, f^*(\tau_0), h^*)$ where $f^*(\tau_0) = \hat{f}(\tau_0, M.\lambda_1)$, $h^* = M.h_1$ are from the first (smallest loss) entry in M;

Set $\tau_{\text{now}} = \tau_0$;

Let T be the list of the fits $\mathbf{f} = (\hat{f}(\tau_0, \lambda_1), ..., \hat{f}(\tau_0, \lambda_m))$, regularization values $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_m)$, rates $\mathbf{c} = (c_1, ..., c_m)$, and directions $\mathbf{h} = (h_1, ..., h_m)$ as defined in Theorem 2; `/* Main loop */`

while $\tau_{\text{now}} < \tau_{\text{end}}$ **do**

  Update $\{\tau_{\text{new}}, k_{\text{new}}, i_{\text{new}}, \text{type}\}$ = FindEvent(T);

  Update $T.\lambda_k = T.\lambda_k + (\tau_{\text{new}} - \tau_{\text{now}})T.c_k$ for $k = 1, ..., m$;

  Update $T.f_k$ according to (6);

  while $\tau_{\text{now}} < \tau_{\text{new}}$ **do**

    Set $\tau_{\text{keep}} = \tau_{\text{now}}$;

    (M, change, τ_{now}) = UpdateValidList($M, \tau_{\text{keep}}, \tau_{\text{new}}$);

    if *change=TRUE* **then**

      $OPT = concatenate(OPT, (\tau_{\text{now}}, f^*(\tau_{\text{now}}), h^*))$ where $f^*(\tau_{\text{now}}) = f(\tau_{\text{now}}, M.\lambda_1)$, $h^* = M.h_1$ are from the first (smallest loss) entry in M;

    end

  end

  if *type = cross* **then** `/* Knot crossing of knots` k_{new}, $k_{\text{new}}+1$ `*/`

    Set T = KnotCross($T, k_{\text{new}}, \tau_{\text{now}}$);

  else if *type = merge* **then** `/* Knot merge of` k_{new}, $k_{\text{new}}+1$, $k_{\text{new}}+2$ `*/`

    Remove knots $k_{\text{new}}, k_{\text{new}+2}$ from T;

    Update sets $\mathcal{E}, \mathcal{R}, \mathcal{L}$ for the remaining knot (Move the observation which defined the two removed knots from $\mathcal{E}$ to $\mathcal{L}$ or $\mathcal{R}$);

    else `/* Knot split of knot` k_{new} `with observation` i_{new} `*/`

      `/* Function` ***KnotSplit*** `would be identical to` ***KnotCross***`---identify two observations at border and find legal directions---except that a split situation yields three such directions, hence three knots, while a cross situation yields two */`

end

Procedure "IntializeValidList": Initialization of bi-level candidate list

Input: Initial value τ_0, vector of knot values $\boldsymbol{\lambda}$, corresponding directions $\mathbf{g}$

Output: A list $M = \{(r_k, \lambda_k, l_k, h_k, \tau_k) : k = 1, ..., m+v\}$ sorted by $l_1 \leq l_2 \leq .... \leq l_{m+v}$, where:

r_k is an indicator in $\{knot, valx\}$ whether this is a knot or a validation crossing

λ_k is its "location" on the path

l_k is its cross validation loss

h_k is its direction

τ_k is knot meeting point

Find knot directions $h_1, ..., h_m$ according to Theorem 2;

/* Identify all validation crossings in the solution path for τ_0 */

Set $V = \Phi$ the empty set;

for $k = 1, ..., m$ *knots and* $i = 1, ..., N$ *validation observations* **do**

  if $\hat{f}(\tau_0, \lambda_{k-1})(\tilde{\mathbf{x}}_i) > \tilde{y}_i$ *and* $\hat{f}(\tau_0, \lambda_k)(\tilde{\mathbf{x}}_i) < \tilde{y}_i$ *or vice versa* **then**

    Set $\tilde{\lambda} = \lambda_k \frac{\hat{f}(\tau_0, \lambda_k)(\tilde{\mathbf{x}}_i) - g_k(\tilde{\mathbf{x}}_i)}{\tilde{y}_i - g_k(\tilde{\mathbf{x}}_i)}$;

    Find the validation crossing direction $\tilde{h}(\mathbf{x})$ according to Proposition 5;

    Add the entry $(\tilde{\lambda}, \tilde{h})$ characterizing the validation crossing to the set V;

  end

end

/* Sort knots and validation crossings by their loss */

Denote the number of validation crossings by $v = |V|$;

for $k = 1, ..., m$ **do**

  Calculate knot validation loss: $l_k = \sum_{i=1}^{N} L_{\tau_0}(\hat{f}(\tau_0, \lambda_k)(\tilde{\mathbf{x}}_i), \tilde{y}_i)$;

end

for $k = 1, ..., v$ **do**

  Calculate validation crossing loss: $l_{m+k} = \sum_{i=1}^{N} L_{\tau_0}(\hat{f}(\tau_0, \tilde{\lambda}_k)(\tilde{\mathbf{x}}_i), \tilde{y}_i)$;

end

Create list $M = \{(r_k, \lambda_k, l_k, h_k, \tau_k) : k = 1, ..., m+v\}$ sorted by $l_1 \leq l_2 \leq .... \leq l_{m+v}$, where:

r_k is an indicator whether this is a knot or a validation crossing with possible values *knot* and *valx* respetively

λ_k is its "location" on the path

l_k is its cross validation loss

h_k is its direction

τ_k is knot meeting point, defined below;

/* Identify mtg pts of neighboring knots or valid. crossings */

for $k = 1, ..., m+v-1$ **do**

  Let $\tau_k = \tau_0 + \delta_k$ where δ_k is the minimal positive solution of the Problem (11) with $l = k+1$;

end

Procedure "`FindEvent`": Find the next event on the path as τ changes

Input: The list T of knots and their directions

Output: Event type in $\{cross, merge, birth\}$, τ_{new} where next event happens, k_{new} the knot where this event happens, i_{new} the observation involved in the event (if a birth)

`/* Next knot crossing or knot merging */`

Set $\tilde{\tau}_k = \frac{T.\lambda_k - T.\lambda_{k+1}}{T.c_{k+1} - T.c_k}$, $k = 1, ..., m-1$;

Set $k_{\text{new}} = \arg\min_{k=1,...,m-1}\{\tilde{\tau}_k : \tilde{\tau}_k > 0\}$;

Set $\tau_{\text{new}} = \tilde{\tau}_{k_{\text{new}}}$ **if** $\tilde{\tau}_{k_{\text{new}}} = \tilde{\tau}_{k_{\text{new}}+1}$ **then**

    type=*merge* ; `/* Knots merging` $3 \Rightarrow 1$ `*/`

else

    type=*cross*; `/* Two knots crossing */`

end

`/* Next observation-knot crossing = knot birth */`

for $k = 1, ..., m$ **do**

    Set $i'_k = \arg\min_{i=1,...,n}\{\frac{T.\lambda_k(\hat{f}(T.\lambda_k)(\mathbf{x}_i) - y_i)}{T.c_k(y_i - h_k(\mathbf{x}_i))} : \frac{T.\lambda_k(\hat{f}(T.\lambda_k)(\mathbf{x}_i) - y_i)}{T.c_k(y_i - h_k(\mathbf{x}_i))} > 0\}$;

    Set τ'_k to be the minimum attained;

end

Set $k' = \arg\min_k \tau'_k$;

if $\tau'_{k'} < \tau_{\text{new}}$ **then**

    Set $\tau_{\text{new}} = \tau'_{k'}$, $k_{\text{new}} = k'$, type=*birth* ;

end

Procedure "UpdateValidList": Find the next validation event on the path as τ changes, and update the list if necessary

Input: Validation candidate list M, current value τ_{keep}, next event on main path τ_{new}

Output: Logical indicator *change* whether optimum changed, Updated list M, and τ_{now} where next validation event happens

Set change = FALSE ;

/* Pair of validation crossings can disappear, or a new a validation crossing can appear, or a regular order change in the elements in M can occur. We first identify the next order change in the list M */

Set $\tau_{\text{now}} = \min_{k=1,\ldots,m+v-1} M.\tau_k$;

Set $k_{\text{now}} = \arg\min_{k=1,\ldots,m+v-1} M.\tau_k$;

/* Now find the next time a validation observation hits a knot $\Rightarrow$ new validation crossing */

For $i = 1,\ldots N$ and $k = 1\ldots,m$ set $\Delta\tau(i,k) = M.\lambda_k \frac{\hat{f}(\tau_{\text{keep}}, M.\lambda_k)(\tilde{\mathbf{x}}_i) - \tilde{y}_i}{\tilde{y}_i M.c_k - M.h_k(\tilde{\mathbf{x}}_i)}$;

Set $\tilde{\tau} = \tau_{\text{keep}} + \min_{i=1,\ldots,N,k=1,\ldots,m}\{\Delta\tau(i,k) : \Delta\tau(i,k) > 0\}$;

Set $(\tilde{i},\tilde{k}) = \arg\min_{i=1,\ldots,N,k=1,\ldots,m}\{\Delta\tau(i,k) : \Delta\tau(i,k) > 0\}$;

if $\tau_{\text{now}} > \tau_{\text{new}}$ *and* $\tilde{\tau} > \tau_{\text{new}}$ **then** /* No validation event before τ_{new} */

  $\tau_{\text{now}} = \tau$new;

  return;

else if $\tau_{\text{now}} > \tilde{\tau}$ **then** /* New validation xing appears---add it to list */

  Set $\tilde{\lambda} = (\tilde{\tau} - \tau_{\text{keep}}) M.c_{\tilde{k}} + M.\lambda_{\tilde{k}}$;

  Set $\tilde{l} = \sum_{i=1}^{N} L_{\tilde{\tau}}(\hat{f}(\tilde{\tau},\tilde{\lambda})(\tilde{\mathbf{x}}_i), \tilde{y}_i)$;

  Set the validation xing direction $\tilde{h}(\mathbf{x})$ according to Proposition 5;

  Find the location k' in the sorted list of the cross validation loss $\tilde{l}$ and insert the element $(r = valx, \tilde{\lambda}, \tilde{l}, \tilde{h})$ into M at location k' ;

  Recalculate $\tau_{k'-1}, \tau_{k'}$ in M according to (11);

  Set $\tau_{\text{now}} = \tilde{\tau}$;

else

  /* If two validation crossings meet knot---the two disappear */

  Set $(merged, M)$ = CheckMerge(M, k_{now});

  if *merged=FALSE* **then** /* Usual situation: swap elements, update meetings */

    Swap elements k_{now} and $k_{\text{now}} + 1$ in M;

    Recalculate $\tau_{k_{\text{now}}-1}, \tau_{k_{\text{now}}}, \tau_{k_{\text{now}}+1}$ in M according to (11);

    if $k_{\text{now}} = 1$ **then** /* First element changed $\Rightarrow$ change of optimum */

      changed = TRUE;

    end

  end

end

Procedure `"CheckMerge"`: Find out if the validation event is in fact two validation crossings of same observation meeting a knot and merging

Input: Validation candidate list M, index of event point k

Output: Logical indicator *merge* whether a merge occurred, updated list M

merge = FALSE;
`/* With observations in general location,` $\tau_k = \tau_{k+1}$ `in` M `implies immediately that we have a merge. Two of` $k, k+1, k+2$ `are validation crossings of the same observation, the knot is the third involved in the crossing. If we do not assume that, more checks are required! */`
if $M.\tau_k = M.\tau_{k+1}$ **then**
    merge = TRUE ;
    `/* Find out which one of the entries` $k, k+1, k+2$ `in` M `is a knot, delete the other two */`
    if $M.r_k =$ knot **then**
        remove entries $k+1, k+2$ from M;
    else if $M.r_{k+1} =$ knot **then**
        remove entries $k, k+2$ from M ;
    else `/*` $M.r_{k+2} =$ knot `*/`
        remove entries $k, k+1$ from M ;
    Update $M.\tau_{k-1}, M.\tau_k$ according to (11);
end

Procedure "KnotCross": Update directions when knots cross

Input: Knot list T, index of first of crossing knots k, current quantile τ

Output: Updated list T

/* Identify i_1, i_2, the ''knot'' observations at the two knots */

Find i_1, i_2 s.t. $\theta_{i_j} \in \{\tau, -(1-\tau)\}$ and $y_{i_j} = \hat{f}(\tau, T.\lambda_{k+j-1})(\mathbf{x}_{i_j})$ for $j \in \{1,2\}$;

Calculate the sets $\mathcal{E}, \mathcal{R}, \mathcal{L}$ as defined in the text for the meeting knots (leaving out the "border observations" i_1, i_2);

Set $u = k$ **for** rel $= 1,2$ **do** /* try releasing each border observation to $\mathcal{E}$ or $\mathcal{L}$ or $\mathcal{R}$ as appropriate */

  Add observation i_{rel} to $\mathcal{E}$ and calculate direction h, c according to Theorem 2;

  /* Check sign and magnitude of $b_{i_{\text{rel}}}$ for consistency (to maintain $\theta_{i_{\text{rel}}} \in [-(1-\tau), \tau]$ as τ increases) */

  if $b_{i_{\text{rel}}} \leq 1$ *and* $\hat{\theta}_{i_{\text{rel}}} = \tau$ **then**

    Update entry u in T with this direction h, c, set $u = u + 1$;

  else if $b_{i_{\text{rel}}} \geq -1$ *and* $\hat{\theta}_{i_{\text{rel}}} = -(1-\tau)$ **then**

    Update entry u in T with this direction h, c, set $u = u + 1$;

  end

  if $\hat{\theta}(i_{\text{rel}}) = \tau$ **then**

    Add i_{rel} to $\mathcal{R}$;

  else /* $\hat{\theta}(i_{\text{rel}}) = -1 - \tau$ */

    Add i_{rel} to $\mathcal{L}$;

  Calculate direction h and c according to Theorem 2;

  /* Check sign and magnitude of $h(\mathbf{x}_{i_{\text{rel}}})$ for sign consistency */

  if $h(\mathbf{x}_{i_{\text{rel}}}) < 0$ *and* $\hat{\theta}_{i_{\text{rel}}} = \tau$ **then**

    Update entry u in T with this direction h, c, set $u = u + 1$;

  else if $h(\mathbf{x}_{i_{\text{rel}}}) > 0$ *and* $\hat{\theta}_{i_{\text{rel}}} = -(1-\tau)$ **then**

    Update entry u in T with this direction h, c, set $u = u + 1$;

  end

end

References

M. Buchinsky. Changes in the u.s. wage structure 1963-1987: Application of quantile regression. *Econometrica*, 62(2):405–458, Mar. 1994.

A. Christmann and I. Steinwart. Consistency of kernel-based quantile regression. *Applied Stochastic Models in Business and Industry*, 24(2):171–183, 2008. ISSN 1524-1904.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32 (2):407–499, 2004.

E. Eide and M.H. Showalter. The effect of school quality on student performance: A quantile regression approach. *Economics Letters*, 58(3):345–350, Mar. 1998.

L. Gunther and J. Zhu. Efficient computation and model selection for the support vector regression. *Neural Computation*, 19(6), 2005.

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

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path of the support vector machine. *JMLR*, 5:1391–1415, Oct 2004.

X. He, P. Ng, and S. Portnoy. Bivariate quantile smoothing splines. *Journal of the Royal Statistical Society, Ser. B*, 60:537–550, 1998.

G. Kimeldorf and G. Wahba. Some results on chebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

R. Koenker. *Quantile Regression*. New York : Cambridge University Press, 2005.

G. Kunapuli, K.P. Bennett, J. Hu, and J.-S. Pang. Bilevel model selection for support vector machines. In Pierre Hansen and Panos Pardalos, editors, *Data Mining and Mathematical Programming [CRM Proceedings and Lecture Notes]*, volume 45. American Mathematical Society, 2008.

Y. Li, Y. Liu, and J. Zhu. Quantile regression in reproducing kernel hilbert spaces. *JASA*, 102(477), 2007.

D. Mease, A.J. Wyner, and A. Buja. Boosted classification trees and class probability/quantile estimation. *JMLR*, 8:409–439, Oct 2007.

N. Meinshausen. Quantile regression forests. *JMLR*, 7:983–999, Jun 2006.

R. K. Pace and R. Barry. Sparse spatial autoregressions. *Statistics and Probability Letters*, 33: 291–297, 1997.

C. Perlich, S. Rosset, R. Lawrence, and B. Zadrozny. High quantile modeling for customer wallet estimation with other applications. In *Proceedings of the Twelfth International Conference on Data Mining, KDD-07*, 2007.

S. Rosset. Bi-level path following for cross validated solution of kernel quantile regression. In *Proceedings of the 25th international conference on Machine Learning*, 2008.

S. Rosset and J. Zhu. Piecewise linear regularized solution paths. *Annals of Statistics*, 35(3), 2007.

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

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

M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and their Geometric Applications*. Cambridge University Press, 1995.

J. Sherman and W.J. Morrison. Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Annals of Mathematical Statistics*, 20:621, 1949.

A.J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

I. Steinwart and A. Christmann. How SVMs can estimate quantiles and the median. In *Neural Information Processing Systems 20*, pages 305–312, 2008.

I. Takeuchi, Q.V. Le, T.D. Sears, and A.J. Smola. Nonparametric quantile estimation. *JMLR*, 7: 1231–1264, Jul 2006.

I. Takeuchi, K. Nomura, and T. Kanamori. Nonparametric conditional density estimation using piecewise-linear solution path of kernel quantile regression. *Neural Computation*, 21(2):533–559, 2009.

G. Wang, D.-Y. Yeung, and F. H. Lochovsky. Two-dimensional solution path for support vector regression. In *Proceedings of the 23rd international conference on Machine learning*, pages 993–1000, 2006.

J. Wang, X. Shen, and Y. Liu. Probability estimation for large margin classifiers. *Biometrika*, 95 (1):149–167, 2008.

M. Yuan. GACV for quantile smoothing splines. *Computational Statistics and Data Analysis*, 5: 813–829, 2006.

H. Zou, T. Hastie, and R. Tibshirani. On the degrees of freedom of the Lasso. *Annals of Statistics*, 35:2173–2192, 2007.

Journal of Machine Learning Research 10 (2009) 2507-2529 Submitted 9/08; Revised 3/09; Published 11/09

When Is There a Representer Theorem? Vector Versus Matrix Regularizers

Andreas Argyriou A.ARGYRIOU@CS.UCL.AC.UK
Department of Computer Science
University College London
Gower Street, London, WC1E 6BT, UK

Charles A. Micchelli CAM@MATH.ALBANY.EDU
Department of Mathematics and Statistics
State University of New York
The University at Albany
Albany, New York 12222, USA

Massimiliano Pontil M.PONTIL@CS.UCL.AC.UK
Department of Computer Science
University College London
Gower Street, London, WC1E 6BT, UK

Editor: Ralph Herbrich

Abstract

We consider a general class of regularization methods which learn a vector of parameters on the basis of linear measurements. It is well known that if the regularizer is a nondecreasing function of the L_2 norm, then the learned vector is a linear combination of the input data. This result, known as the *representer theorem*, lies at the basis of kernel-based methods in machine learning. In this paper, we prove the necessity of the above condition, in the case of differentiable regularizers. We further extend our analysis to regularization methods which learn a matrix, a problem which is motivated by the application to multi-task learning. In this context, we study a more general representer theorem, which holds for a larger class of regularizers. We provide a necessary and sufficient condition characterizing this class of matrix regularizers and we highlight some concrete examples of practical importance. Our analysis uses basic principles from matrix theory, especially the useful notion of matrix nondecreasing functions.

Keywords: kernel methods, matrix learning, minimal norm interpolation, multi-task learning, regularization

1. Introduction

Regularization in Hilbert spaces is an important methodology for learning from examples and has a long history in a variety of fields. It has been studied, from different perspectives, in statistics (see Wahba, 1990, and references therein), in optimal estimation (Micchelli and Rivlin, 1985) and recently has been a focus of attention in machine learning theory, see, for example (Cucker and Smale, 2001; De Vito et al., 2004; Micchelli and Pontil, 2005a; Shawe-Taylor and Cristianini, 2004; Vapnik, 2000) and references therein. Regularization is formulated as an *optimization problem* involving an *error term* and a *regularizer*. The regularizer plays an important role, in that it favors solutions with certain desirable properties. It has long been observed that certain regularizers exhibit an appealing

property, called the *representer theorem*, which states that there exists a solution of the regularization problem that is a linear combination of the data. This property has important computational implications in the context of regularization with positive semidefinite *kernels*, because it transforms high or infinite-dimensional problems of this type into finite dimensional problems of the size of the number of available data (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004).

The topic of interest in this paper will be to determine the conditions under which representer theorems hold. In the first half of the paper, we describe a property which a regularizer should satisfy in order to give rise to a representer theorem. It turns out that this property has a simple geometric interpretation and that the regularizer can be equivalently expressed as a *nondecreasing* function of the Hilbert space norm. Thus, we show that this condition, which has already been known to be sufficient for representer theorems, is also *necessary*. In the second half of the paper, we depart from the context of Hilbert spaces and focus on a class of problems in which *matrix structure* plays an important role. For such problems, which have recently appeared in several machine learning applications, we show a modified version of the representer theorem that holds for a class of regularizers significantly larger than in the former context. As we shall see, these matrix regularizers are important in the context of multi-task learning: the matrix columns are the parameters of different regression tasks and the regularizer encourages certain dependences across the tasks.

In general, we consider problems in the framework of *Tikhonov regularization* (Tikhonov and Arsenin, 1977). This approach finds, on the basis of a set of input/output data $(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{H} \times \mathcal{Y}$, a vector in $\mathcal{H}$ as the solution of an optimization problem. Here, $\mathcal{H}$ is a prescribed Hilbert space equipped with the inner product $\langle \cdot, \cdot \rangle$ and $\mathcal{Y} \subseteq \mathbb{R}$ a set of possible output values. The optimization problems encountered in regularization are of the type

$$\min \left\{ \mathcal{E}\big((\langle w, x_1 \rangle, \ldots, \langle w, x_m \rangle), (y_1, \ldots, y_m) \big) + \gamma\, \Omega(w) : w \in \mathcal{H} \right\}, \tag{1}$$

where $\gamma > 0$ is a regularization parameter. The function $\mathcal{E} : \mathbb{R}^m \times \mathcal{Y}^m \to \mathbb{R}$ is called an *error function* and $\Omega : \mathcal{H} \to \mathbb{R}$ is called a *regularizer*. The error function measures the error on the data. Typically, it decomposes as a sum of univariate functions. For example, in regression, a common choice would be the sum of square errors, $\sum_{i=1}^m (\langle w, x_i \rangle - y_i)^2$. The function Ω, called the regularizer, favors certain regularity properties of the vector w (such as a small norm) and can be chosen based on available prior information about the target vector. In some Hilbert spaces such as Sobolev spaces the regularizer is a measure of smoothness: the smaller the norm the smoother the function.

This framework includes several well-studied learning algorithms, such as ridge regression (Hoerl and Kennard, 1970), support vector machines (Boser et al., 1992), and many more—see Schölkopf and Smola (2002) and Shawe-Taylor and Cristianini (2004) and references therein.

An important aspect of the practical success of this approach is the observation that, for certain choices of the regularizer, solving (1) reduces to identifying m parameters and not $\dim(\mathcal{H})$. Specifically, when the regularizer is the square of the Hilbert space norm, the representer theorem holds: there exists a solution $\hat{w}$ of (1) which is a linear combination of the input vectors,

$$\hat{w} = \sum_{i=1}^m c_i x_i, \tag{2}$$

where c_i are some real coefficients. This result is simple to prove and dates at least from the 1970's, see, for example, Kimeldorf and Wahba (1970). It is also known that it extends to any regularizer that is a *nondecreasing* function of the norm (Schölkopf et al., 2001). Several other variants

and results about the representation form (2) have also appeared in recent years (De Vito et al., 2004; Dinuzzo et al., 2007; Evgeniou et al., 2000; Girosi et al., 1995; Micchelli and Pontil, 2005b; Steinwart, 2003; Wahba, 1992). Moreover, the representer theorem has been important in machine learning, particularly within the context of learning in reproducing kernel Hilbert spaces (Aronszajn, 1950)—see Schölkopf and Smola (2002) and Shawe-Taylor and Cristianini (2004) and references therein.

Our first objective in this paper is to derive necessary and sufficient conditions for representer theorems to hold. Even though one is mainly interested in regularization problems, it is more convenient to study *interpolation* problems, that is, problems of the form

$$\min\left\{\Omega(w) : w \in \mathcal{H}, \langle w, x_i\rangle = y_i,\ \forall i = 1,\ldots,m\right\}. \tag{3}$$

Thus, we begin this paper (Section 2) by showing how representer theorems for interpolation and regularization relate. On one side, a representer theorem for interpolation easily implies such a theorem for regularization with the same regularizer and any error function. Therefore, *all representer theorems obtained in this paper apply equally to interpolation and regularization*. On the other side, though, the converse implication is true under certain weak qualifications on the error function.

Having addressed this issue, we concentrate in Section 3 on proving that an interpolation problem (3) admits solutions representable in the form (2) *if and only if* the regularizer is *a nondecreasing function of the Hilbert space norm*. That is, we provide a complete characterization of regularizers that give rise to representer theorems, which had been an open question. Furthermore, we discuss how our proof is motivated by a geometric understanding of the representer theorem, which is equivalently expressed as a monotonicity property of the regularizer. We note that for simplicity throughout the paper we shall assume that Ω is differentiable. However our results are constructive and it should be possible to extend them to the non-differentiable case.

Our second objective is to formulate and study the novel question of representer theorems for *matrix problems*. To make our discussion concrete, let us consider the problem of learning n linear regression vectors, represented by the parameters $w_1,\ldots,w_n \in \mathbb{R}^d$, respectively. Each vector can be thought of as a "task" and the goal is to *jointly* learn these n tasks. In such problems, there is usually prior knowledge that *relates* these tasks and it is often the case that learning can improve if this knowledge is appropriately taken into account. Consequently, a good regularizer should favor such task relations and involve *all tasks jointly*.

In the case of interpolation, this learning framework can be formulated concisely as

$$\min\left\{\Omega(W) : W \in \mathbf{M}_{d,n},\ w_t^\top x_{ti} = y_{ti},\ \ \forall i = 1,\ldots,m_t,\ t = 1,\ldots,n\right\}, \tag{4}$$

where $\mathbf{M}_{d,n}$ denotes the set of $d \times n$ real matrices and the column vectors $w_1,\ldots,w_n \in \mathbb{R}^d$ form the matrix W. Each task t has its own input data $x_{t1},\ldots,x_{tm_t} \in \mathbb{R}^d$ and corresponding output values $y_{t1},\ldots,y_{tm_t} \in \mathcal{Y}$.

An important feature of such problems that distinguishes them from the type (3) is the appearance of *matrix products* in the constraints, unlike the inner products in (3). In fact, as we will discuss in Section 4.1, problems of the type (4) can be written in the form (3). Consequently, the representer theorem applies if the matrix regularizer is a nondecreasing function of the Frobenius norm.[1] However, the optimal vector $\hat{w}_t$ for each task can be represented as a linear combination of

1. Defined as $\|W\|_2 = \sqrt{\mathrm{tr}(W^\top W)}$.

only those input vectors corresponding to this particular task. Moreover, with such regularizers it is easy to see that each task in (4) can be optimized independently. Hence, these regularizers are of no practical interest if the tasks are expected to be related.

This observation leads us to formulate a *modified representer theorem*, which is appropriate for matrix problems, namely,

$$\hat{w}_t = \sum_{s=1}^{n} \sum_{i=1}^{m_s} c_{si}^{(t)} x_{si} \qquad \forall t = 1, \ldots, n, \tag{5}$$

where $c_{si}^{(t)}$ are scalar coefficients, for $t, s = 1, \ldots, n,\ i = 1, \ldots, m_s$. In other words, we now allow for *all input vectors* to be present in the linear combination representing each column of the optimal matrix. As a result, this definition greatly expands the class of regularizers that give rise to representer theorems.

Moreover, this framework can be used in many applications where matrix optimization problems are involved. Our immediate motivation, however, has been more specific than that, namely *multi-task learning*. Learning multiple tasks jointly has been a growing area of interest in machine learning and other fields, especially during the past few years (Abernethy et al., 2009; Argyriou et al., 2006, 2008a, 2007; Candès and Recht, 2009; Cavallanti et al., 2008; Izenman, 1975; Maurer, 2006a,b; Srebro et al., 2005; Wolf et al., 2007; Xiang and Bennett, 2005; Xiong et al., 2006; Yuan et al., 2007). For instance, some of these approaches use regularizers which involve the *trace norm*[2] of matrix W. There have been several motivations for the trace norm, derived from kernel learning, maximum likelihood, matrix factorization or graphical models. Another motivation is that under conditions a small trace norm favors low-rank matrices. This means that the tasks (the columns of W) are related in that they all lie in a low-dimensional subspace of $\mathbb{R}^d$. In the case of the trace norm, the representer theorem (5) is known to hold—see Abernethy et al. (2009), Argyriou et al. (2008a) and Amit et al. (2007); see also the discussion in Section 4.1.

It is natural, therefore, to ask a question similar to that in the standard Hilbert space (or single-task) setting. That is, under which conditions on the regularizer a representer theorem holds. In Section 4.2, we provide an answer by *proving a necessary and sufficient condition for representer theorems* (5) *to hold for problem* (4)*, expressed as a simple monotonicity property*. This property is analogous to the one in the Hilbert space setting, but its geometric interpretation is now algebraic in nature. We also give a functional description equivalent to this property, that is, *we show that the regularizers of interest are the matrix nondecreasing functions of the quantity* $W^\top W$.

Our results cover matrix problems of the type (4) which have already been studied in the literature. But they also point towards some new learning methods that may perform well in practice and can now be made computationally efficient. Thus, we close the paper with a discussion of possible regularizers that satisfy our conditions and have been used or can be used in the future in machine learning problems.

1.1 Notation

Before proceeding, we introduce the notation used in this paper. We use $\mathbb{N}_d$ as a shorthand for the set of integers $\{1, \ldots, d\}$. We use $\mathbb{R}^d$ to denote the linear space of vectors with d real components. The standard inner product in this space is denoted by $\langle \cdot, \cdot \rangle$, that is, $\langle w, v \rangle = \sum_{i \in \mathbb{N}_d} w_i v_i,\ \forall w, v \in \mathbb{R}^d$,

2. Equal to the sum of the singular values of W.

where w_i, v_i are the i-th components of w, v respectively. More generally, we will consider Hilbert spaces which we will denote by $\mathcal{H}$, equipped with an inner product $\langle \cdot, \cdot \rangle$.

We also let $\mathbf{M}_{d,n}$ be the linear space of $d \times n$ real matrices. If $W, Z \in \mathbf{M}_{d,n}$ we define their Frobenius inner product as $\langle W, Z \rangle = \mathrm{tr}(W^\top Z)$, where tr denotes the trace of a matrix. With $\mathbf{S}^d$ we denote the set of $d \times d$ real symmetric matrices and with $\mathbf{S}^d_+$ ($\mathbf{S}^d_{++}$) its subset of positive semidefinite (definite) ones. We use $\succ$ and $\succeq$ for the positive definite and positive semidefinite partial orderings, respectively. Finally, we let $\mathbf{O}^d$ be the set of $d \times d$ orthogonal matrices.

2. Regularization Versus Interpolation

The line of attack which we shall follow in this paper will go through *interpolation*. That is, our main concern will be to obtain necessary and sufficient conditions for representer theorems that hold for interpolation problems. However, in practical applications one encounters *regularization* problems more frequently than interpolation problems.

First of all, the family of the former problems is more general than that of the latter ones. Indeed, an interpolation problem can be simply obtained in the limit as the *regularization parameter* goes to zero (Micchelli and Pinkus, 1994). More importantly, regularization enables one to trade off interpolation of the data against smoothness or simplicity of the model, whereas interpolation frequently suffers from *overfitting*.

Thus, frequently one considers problems of the form

$$\min \left\{ \mathcal{E}\big((\langle w, x_1 \rangle, \ldots, \langle w, x_m \rangle), (y_1, \ldots, y_m) \big) + \gamma\, \Omega(w) : w \in \mathcal{H} \right\}, \tag{6}$$

where $\gamma > 0$ is called the regularization parameter. This parameter is not known in advance but can be tuned with techniques like *cross validation*, see, for example, Wahba (1990). Here, the function $\Omega : \mathcal{H} \to \mathbb{R}$ is a *regularizer*, $\mathcal{E} : \mathbb{R}^m \times \mathcal{Y}^m \to \mathbb{R}$ is an error function and $x_i \in \mathcal{H}, y_i \in \mathcal{Y}, \forall i \in \mathbb{N}_m$, are given input and output data. The set $\mathcal{Y}$ is a subset of $\mathbb{R}$ and varies depending on the context, so that it is typically assumed equal to $\mathbb{R}$ in the case of regression or equal to $\{-1, 1\}$ in binary classification problems. One may also consider the associated interpolation problem, which is

$$\min \left\{ \Omega(w) : w \in \mathcal{H}, \langle w, x_i \rangle = y_i,\ \forall i \in \mathbb{N}_m \right\}. \tag{7}$$

Under certain assumptions, the minima in problems (6) and (7) are attained (the latter whenever the constraints in (7) are satisfiable). Such assumptions could involve, for example, lower semicontinuity and boundedness of sublevel sets for Ω and boundedness from below for $\mathcal{E}$. These issues will not concern us here, as we shall assume the following about the error function $\mathcal{E}$ and the regularizer Ω, from now on.

Assumption 1 *The minimum* (6) *is attained for any* $\gamma > 0$, *any input and output data* $\{x_i, y_i : i \in \mathbb{N}_m\}$ *and any* $m \in \mathbb{N}$. *The minimum* (7) *is attained for any input and output data* $\{x_i, y_i : i \in \mathbb{N}_m\}$ *and any* $m \in \mathbb{N}$, *whenever the constraints in* (7) *are satisfiable*.

The main objective of this paper is to obtain *necessary and sufficient* conditions on Ω so that the solution of problem (6) satisfies a *linear representer theorem*.

Definition 2 *We say that a class of optimization problems such as* (6) *or* (7) *satisfies the* linear representer theorem *if, for any choice of data* $\{x_i, y_i : i \in \mathbb{N}_m\}$ *such that the problem has a solution,* there exists *a solution that belongs to* $\mathrm{span}\{x_i : i \in \mathbb{N}_m\}$.

In this section, we show that the existence of representer theorems for regularization problems is equivalent to the existence of representer theorems for interpolation problems, under a quite general condition that has a simple geometric interpretation.

We first recall a lemma from (Micchelli and Pontil, 2004, Sec. 2) which states that (linear or not) representer theorems for interpolation lead to representer theorems for regularization, under no conditions on the error function.

Lemma 3 *Let $E : \mathbb{R}^m \times Y^m \to \mathbb{R}$, $\Omega : \mathcal{H} \to \mathbb{R}$ satisfying Assumption 1. Then if the class of interpolation problems* (7) *satisfies the linear representer theorem, so does the class of regularization problems* (6).

Proof Consider a problem of the form (6) and let $\hat{w}$ be a solution. We construct an associated interpolation problem

$$\min\left\{\Omega(w) : w \in \mathcal{H}, \langle w, x_1\rangle = \langle \hat{w}, x_1\rangle, \ldots, \langle w, x_m\rangle = \langle \hat{w}, x_m\rangle\right\}. \tag{8}$$

By hypothesis, there exists a solution $\tilde{w}$ of (8) that lies in $\operatorname{span}\{x_i : i \in \mathbb{N}_m\}$. But then $\Omega(\tilde{w}) \leq \Omega(\hat{w})$ and hence $\tilde{w}$ is a solution of (6). The result follows. ■

This lemma requires no special properties of the functions involved. Its converse, in contrast, requires assumptions about the analytical properties of the error function. We provide one such natural condition in the theorem below, but other conditions could conceivably work too. The main idea in the proof is, based on a single input, to construct a sequence of appropriate regularization problems for different values of the regularization parameter γ. Then, it suffices to show that letting $\gamma \to 0^+$ yields a limit of the minimizers that satisfies an interpolation constraint.

Theorem 4 *Let $E : \mathbb{R}^m \times Y^m \to \mathbb{R}$ and $\Omega : \mathcal{H} \to \mathbb{R}$. Assume that E, Ω are lower semicontinuous, that Ω has bounded sublevel sets and that E is bounded from below. Assume also that, for some $v \in \mathbb{R}^m \setminus \{0\}, y \in Y^m$, there exists a* unique *minimizer of* $\min\{E(av, y) : a \in \mathbb{R}\}$ *and that this minimizer does not equal zero. Then if the class of regularization problems* (6) *satisfies the linear representer theorem, so does the class of interpolation problems* (7).

Proof Fix an arbitrary $x \neq 0$ and let a_0 be the minimizer of $\min\{E(av, y) : a \in \mathbb{R}\}$. Consider the problems

$$\min\left\{E\left(\frac{a_0}{\|x\|^2}\langle w, x\rangle v, y\right) + \gamma\,\Omega(w) : w \in \mathcal{H}\right\},$$

for every $\gamma > 0$, and let w_γ be a solution in the span of x (known to exist by hypothesis). We then obtain that

$$E(a_0 v, y) + \gamma\Omega(w_\gamma) \leq E\left(\frac{a_0}{\|x\|^2}\langle w_\gamma, x\rangle v, y\right) + \gamma\,\Omega(w_\gamma) \leq E(a_0 v, y) + \gamma\,\Omega(x). \tag{9}$$

Thus, $\Omega(w_\gamma) \leq \Omega(x)$ and so, by the hypothesis on Ω, the set $\{w_\gamma : \gamma > 0\}$ is bounded. Therefore, there exists a convergent subsequence $\{w_{\gamma_\ell} : \ell \in \mathbb{N}\}$, with $\gamma_\ell \to 0^+$, whose limit we call $\bar{w}$. By taking the limit inferior as $\ell \to \infty$ on the inequality on the right in (9), we obtain

$$E\left(\frac{a_0}{\|x\|^2}\langle \bar{w}, x\rangle v, y\right) \leq E(a_0 v, y).$$

Consequently,

$$\frac{a_0}{\|x\|^2}\langle \bar{w}, x\rangle \;= a_0$$

or, using the hypothesis that $a_0 \neq 0$,

$$\langle \bar{w}, x\rangle \;= \|x\|^2.$$

In addition, since w_γ belongs to the span of x for every $\gamma > 0$, so does $\bar{w}$. Thus, we obtain that $\bar{w} = x$. Moreover, from the definition of w_γ we have that

$$\mathcal{E}\left(\frac{a_0}{\|x\|^2}\langle w_\gamma, x\rangle\, v, y\right) + \gamma\,\Omega(w_\gamma) \leq \mathcal{E}\,(a_0\, v, y) + \gamma\,\Omega(w) \qquad \forall w \in \mathcal{H} \text{ such that } \langle w, x\rangle = \|x\|^2$$

and, combining with the definition of a_0, we obtain that

$$\Omega(w_\gamma) \leq \Omega(w) \qquad \forall w \in \mathcal{H} \text{ such that } \langle w, x\rangle = \|x\|^2.$$

Taking the limits inferior as $\ell \to \infty$, we conclude that $\bar{w} = x$ is a solution of the problem

$$\min\{\Omega(w) : w \in \mathcal{H}, \langle w, x\rangle = \|x\|^2\}\,.$$

Moreover, this assertion holds even when $x = 0$, since the hypothesis implies that 0 is a global minimizer of Ω. Indeed, any regularization problem of the type (6) with zero inputs, $x_i = 0, \forall i \in \mathbb{N}_m$, admits a solution in their span. Thus, we have shown that Ω satisfies property (13) in the next section and the result follows immediately from Lemma 9 below. ■

We now comment on some commonly used error functions. The first is the *square loss*,

$$\mathcal{E}(z, y) = \sum_{i \in \mathbb{N}_m} (z_i - y_i)^2\,,$$

for $z, y \in \mathbb{R}^m$. It is immediately apparent that Theorem 4 applies in this case.

The second function is the *hinge loss*,

$$\mathcal{E}(z, y) = \sum_{i \in \mathbb{N}_m} \max(1 - z_i y_i, 0)\,,$$

where the outputs y_i are assumed to belong to $\{-1, 1\}$ for the purpose of classification. In this case, we may select $y_i = 1, \forall i \in \mathbb{N}_m$, and $v = (1, -2, 0, \ldots, 0)^\top$ for $m \geq 2$. Then the function $\mathcal{E}(\cdot\, v, y)$ is the one shown in Figure 1.

Finally, the *logistic loss*,

$$\mathcal{E}(z, y) = \sum_{i \in \mathbb{N}_m} \log\left(1 + e^{-z_i y_i}\right)\,,$$

is also used in classification problems. In this case, we may select $y_i = 1, \forall i \in \mathbb{N}_m$, and $v = (2, -1)^\top$ for $m = 2$ or $v = (m-2, -1, \ldots, -1)^\top$ for $m > 2$. In the latter case, for example, setting to zero the derivative of $\mathcal{E}(\cdot\, v, y)$ yields the equation $(m-1)e^{a(m-1)} + e^a - m + 2 = 0$, which can easily be seen to have a unique solution.

Summarizing, we obtain the following corollary.

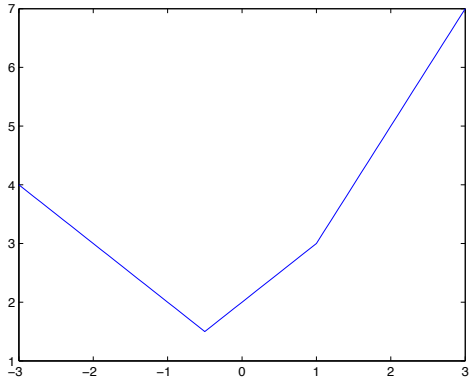


Figure 1: Hinge loss along the direction $(1,-2,0,\ldots,0)$.

Corollary 5 *If $\mathcal{E}:\mathbb{R}^m\times\mathcal{Y}^m\to\mathbb{R}$ is the square loss, the hinge loss (for $m\geq 2$) or the logistic loss (for $m\geq 2$) and $\Omega:\mathcal{H}\to\mathbb{R}$ is lower semicontinuous with bounded sublevel sets, then the class of problems* (6) *satisfies the linear representer theorem if and only if the class of problems* (7) *does.*

Note also that the condition on $\mathcal{E}$ in Theorem 4 is rather weak in that an error function $\mathcal{E}$ may satisfy it without being convex. At the same time, an error function that is "too flat", such as a constant loss, will not do.

We conclude with a remark about the situation in which the inputs x_i are *linearly independent*.[3] It has a brief and straightforward proof, which we do not present here.

Remark 6 *Let $\mathcal{E}$ be the hinge loss or the logistic loss and $\Omega:\mathcal{H}\to\mathbb{R}$ be of the form $\Omega(w)=h(\|w\|)$, where $h:\mathbb{R}_+\to\mathbb{R}$ is a lower semicontinuous function with bounded sublevel sets. Then the class of regularization problems* (6) *in which the inputs $x_i, i\in\mathbb{N}_m$, are linearly independent, satisfies the linear representer theorem.*

3. Representer Theorems for Interpolation Problems

The results of the previous section allow us to focus on linear representer theorems for interpolation problems of the type (7). We are going to consider the case of a Hilbert space $\mathcal{H}$ as the domain of an interpolation problem. Interpolation constraints will be formed as inner products of the variable with the input data.

In this section, we consider the interpolation problem

$$\min\{\Omega(w): w\in\mathcal{H}, \langle w,x_i\rangle = y_i, i\in\mathbb{N}_m\}. \tag{10}$$

We coin the term *admissible* to denote the class of regularizers we are interested in.

Definition 7 *We say that the function $\Omega:\mathcal{H}\to\mathbb{R}$ is* admissible *if, for every $m\in\mathbb{N}$ and any data set $\{(x_i,y_i): i\in\mathbb{N}_m\}\subseteq\mathcal{H}\times\mathcal{Y}$ such that the interpolation constraints are satisfiable, problem* (10)

3. This occurs frequently in practice, especially when the dimensionality d is high.

admits a solution $\hat{w}$ of the form

$$\hat{w} = \sum_{i \in \mathbb{N}_m} c_i x_i, \tag{11}$$

where c_i are some real parameters.

We say that $\Omega : \mathcal{H} \to \mathbb{R}$ is differentiable if, for every $w \in \mathcal{H}$, there is a unique vector denoted by $\nabla\Omega(w) \in \mathcal{H}$, such that for all $p \in \mathcal{H}$,

$$\lim_{t \to 0} \frac{\Omega(w + tp) - \Omega(w)}{t} = \langle \nabla\Omega(w), p \rangle.$$

This notion corresponds to the usual notion of directional derivative on $\mathbb{R}^d$ and in that case $\nabla\Omega(w)$ is the gradient of Ω at w.

In the remainder of the section, we always assume that Assumption 1 holds for Ω. The following theorem provides a necessary and sufficient condition for a regularizer to be admissible.

Theorem 8 *Let $\Omega : \mathcal{H} \to \mathbb{R}$ be a differentiable function and $\dim(\mathcal{H}) \geq 2$. Then Ω is admissible if and only if*

$$\Omega(w) = h(\langle w, w \rangle) \qquad \forall\, w \in \mathcal{H}, \tag{12}$$

for some nondecreasing function $h : \mathbb{R}_+ \to \mathbb{R}$.

It is well known that the above functional form is sufficient for a representer theorem to hold, see, for example, Schölkopf et al. (2001). Here we show that it is also necessary.

The route we follow to prove the above theorem is based on a geometric interpretation of representer theorems. This intuition can be formally expressed as condition (13) in the lemma below. Both condition (13) and functional form (12) express the property that the contours of Ω are *spheres* (or regions between spheres), which is apparent from Figure 2.

Lemma 9 *A function $\Omega : \mathcal{H} \to \mathbb{R}$ is admissible if and only if it satisfies the property that*

$$\Omega(w + p) \geq \Omega(w) \qquad \forall\, w, p \in \mathcal{H} \text{ such that } \langle w, p \rangle = 0. \tag{13}$$

Proof Suppose that Ω satisfies property (13), consider arbitrary data $x_i, y_i, i \in \mathbb{N}_m$, and let $\hat{w}$ be a solution to problem (10). We can uniquely decompose $\hat{w}$ as $\hat{w} = \bar{w} + p$ where $\bar{w} \in \mathcal{L} := \text{span}\{x_i : i \in \mathbb{N}_m\}$ and $p \in \mathcal{L}^\perp$. From (13) we obtain that $\Omega(\hat{w}) \geq \Omega(\bar{w})$. Also $\bar{w}$ satisfies the interpolation constraints and hence we conclude that $\bar{w}$ is a solution to problem (10).

Conversely, if Ω is admissible choose any $w \in \mathcal{H}$ and consider the problem $\min\{\Omega(z) : z \in \mathcal{H}, \langle z, w \rangle = \langle w, w \rangle\}$. By hypothesis, there exists a solution belonging in $\text{span}\{w\}$ and hence w is a solution to this problem. Thus, we have that $\Omega(w + p) \geq \Omega(w)$ for every p such that $\langle w, p \rangle = 0$. ■

It remains to establish the equivalence of the geometric property (13) to condition (12), which says that Ω is a nondecreasing function of the L_2 norm.

Proof of Theorem 8 Assume first that (13) holds and $\dim(\mathcal{H}) < \infty$. In this case, we only need to consider the case that $\mathcal{H} = \mathbb{R}^d$ since (13) can always be rewritten as an equivalent condition on $\mathbb{R}^d$, using an orthonormal basis of $\mathcal{H}$.

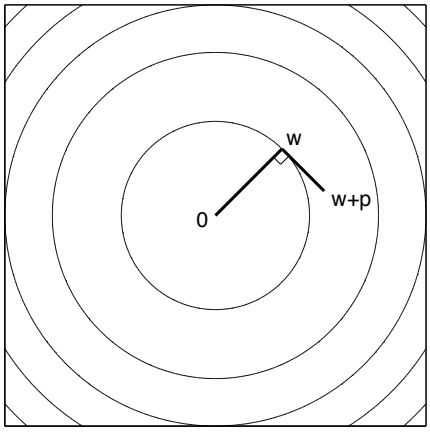


Figure 2: Geometric interpretation of Theorem 8. The function Ω should not decrease when moving to orthogonal directions. The contours of such a function should be spherical.

First we observe that, since Ω is differentiable, this property implies the condition that

$$\langle \nabla \Omega(w), p \rangle = 0\,, \tag{14}$$

for all $w, p \in \mathbb{R}^d$ such that $\langle w, p \rangle = 0$.

Now, fix any $w_0 \in \mathbb{R}^d$ such that $\|w_0\| = 1$. Consider an arbitrary $w \in \mathbb{R}^d \setminus \{0\}$. Then there exists an orthogonal matrix $U \in \mathbf{O}^d$ such that $w = \|w\| U w_0$ and $\det(U) = 1$ (see Lemma 20 in the appendix). Moreover, we can write $U = e^D$ for some skew-symmetric matrix $D \in \mathbf{M}_{d,d}$ (see Example 6.2.15 in Horn and Johnson, 1991). Consider now the path $z : [0,1] \to \mathbb{R}^d$ with

$$z(\lambda) = \|w\| e^{\lambda D} w_0 \qquad\qquad \forall\, \lambda \in [0,1].$$

We have that $z(0) = \|w\| w_0$ and $z(1) = w$. Moreover, since $\langle z(\lambda), z(\lambda) \rangle = \langle w, w \rangle$, we obtain that

$$\langle z'(\lambda), z(\lambda) \rangle = 0 \qquad\qquad \forall\, \lambda \in (0,1).$$

Applying (14) with $w = z(\lambda), p = z'(\lambda)$, it follows that

$$\frac{d\Omega(z(\lambda))}{d\lambda} = \langle \nabla \Omega(z(\lambda)), z'(\lambda) \rangle = 0.$$

Consequently, $\Omega(z(\lambda))$ is constant and hence $\Omega(w) = \Omega(\|w\| w_0)$. Setting $h(\xi) = \Omega(\sqrt{\xi} w_0),\ \forall \xi \in \mathbb{R}_+$, yields (12). In addition, h must be nondecreasing in order for Ω to satisfy property (13).

For the case $\dim(\mathcal{H}) = \infty$ we can argue similarly using instead the path

$$z(\lambda) = \frac{(1-\lambda) w_0 + \lambda w}{\|(1-\lambda) w_0 + \lambda w\|} \|w\|$$

which is differentiable on $(0,1)$ when $w \notin \text{span}\{w_0\}$. We confirm equation (12) for vectors in $\text{span}\{w_0\}$ by a limiting argument on vectors not in $\text{span}\{w_0\}$ since Ω is continuous.

Conversely, if $\Omega(w) = h(\langle w, w\rangle)$ and h is nondecreasing, property (13) follows immediately. ■

The assumption of differentiability is crucial for the above proof and we postpone the issue of removing it to future work. Nevertheless, property (13) follows immediately from functional form (12) without any assumptions.

Remark 10 *Let $\Omega : \mathcal{H} \to \mathbb{R}$ be a function of the form*

$$\Omega(w) = h(\langle w, w\rangle) \qquad \forall\, w \in \mathcal{H},$$

for some nondecreasing function $h : \mathbb{R}_+ \to \mathbb{R}$. Then Ω is admissible.

We also note that we could modify Definition 7 by requiring that *any* solution of problem (10) be in the linear span of the input data. We call such regularizers *strictly admissible*. Then with minor modifications to Lemma 9 (namely, requiring that equality in (13) holds only if $p = 0$) and to the proof of Theorem 8 (namely, requiring h to be strictly increasing) we have the following corollary.

Corollary 11 *Let $\Omega : \mathcal{H} \to \mathbb{R}$ be a differentiable function and $dim(\mathcal{H}) \geq 2$. Then Ω is* strictly *admissible if and only if $\Omega(w) = h(\langle w, w\rangle)$, $\forall w \in \mathcal{H}$, where $h : \mathbb{R}_+ \to \mathbb{R}$ is* strictly *increasing.*

Theorem 8 can be used to verify whether the linear representer theorem can be obtained when using a regularizer Ω. For example, the function $\|w\|_p^p = \sum_{i \in \mathbb{N}_d} |w_i|^p$ is not admissible for any $p > 1, p \neq 2$, because it cannot be expressed as a function of the Hilbert space norm. Indeed, if we choose any $a \in \mathbb{R}$ and let $w = (a\delta_{i1} : i \in \mathbb{N}_d)$, the requirement that $\|w\|_p^p = h(\langle w, w\rangle)$ would imply that $h(a^2) = |a|^p, \forall a \in \mathbb{R}$, and hence that $\|w\|_p = \|w\|$.

4. Matrix Learning Problems

In this section, we investigate how representer theorems and results like Theorem 8 can be extended in the context of optimization problems which involve matrices.

4.1 Exploiting Matrix Structure

As we have already seen, our discussion in Section 3 applies to any Hilbert space. Thus, we may consider the finite Hilbert space of $d \times n$ matrices $\mathbf{M}_{d,n}$ equipped with the Frobenius inner product $\langle \cdot, \cdot \rangle$. As in Section 3, we could consider interpolation problems of the form

$$\min \{\Omega(W) : W \in \mathbf{M}_{d,n}, \langle W, X_i\rangle = y_i, \forall i \in \mathbb{N}_m\} \tag{15}$$

where $X_i \in \mathbf{M}_{d,n}$ are prescribed input matrices and $y_i \in \mathcal{Y}$ are scalar outputs, for $i \in \mathbb{N}_m$. Then Theorem 8 states that such a problem admits a solution of the form

$$\hat{W} = \sum_{i \in \mathbb{N}_m} c_i X_i, \tag{16}$$

where c_i are some real parameters, if and only if Ω can be written in the form

$$\Omega(W) = h(\langle W, W\rangle) \qquad \forall\, W \in \mathbf{M}_{d,n}, \tag{17}$$

where $h : \mathbb{R}_+ \to \mathbb{R}$ is nondecreasing.

However, in machine learning practice, optimization problems of the form (15) occur most frequently in a more special form. That is, usually the constraints of (15) use the structure inherent in matrices—and hence the matrix variable cannot be regarded as a vector variable. Thus, in many recent applications, some of which we shall briefly discuss below, it is natural to consider problems like

$$\min\{\Omega(W) : W \in \mathbf{M}_{d,n},\ w_t^\top x_{ti} = y_{ti},\ \ \forall i \in \mathbb{N}_{m_t}, t \in \mathbb{N}_n\}\,. \tag{18}$$

Here, $w_t \in \mathbb{R}^d$ denote the columns of matrix W, for $t \in \mathbb{N}_n$, and $x_{ti} \in \mathbb{R}^d, y_{ti} \in \mathcal{Y}$ are prescribed inputs and outputs, for $i \in \mathbb{N}_{m_t}, t \in \mathbb{N}_n$. In addition, the desired representation form for solutions of such matrix problems is different from (16). In this case, one may encounter representer theorems of the form

$$\hat{w}_t = \sum_{s \in \mathbb{N}_n} \sum_{i \in \mathbb{N}_{m_s}} c_{si}^{(t)} x_{si} \qquad\qquad \forall t \in \mathbb{N}_n, \tag{19}$$

where $c_{si}^{(t)}$ are scalar coefficients for $s, t \in \mathbb{N}_n, i \in \mathbb{N}_{m_s}$.

To illustrate the above, consider the problem of multi-task learning and problems closely related to it (Abernethy et al., 2009; Ando and Zhang, 2005; Argyriou et al., 2006, 2008a, 2007; Candès and Recht, 2009; Cavallanti et al., 2008; Izenman, 1975; Maurer, 2006a,b; Raina et al., 2006; Srebro et al., 2005; Xiong et al., 2006; Yuan et al., 2007, etc.). In learning multiple tasks jointly, each task may be represented by a vector of regression parameters that corresponds to the column w_t in our notation. There are n tasks and m_t data examples $\{(x_{ti}, y_{ti}) : i \in \mathbb{N}_{m_t}\}$ for the t-th task. The learning algorithm used is

$$\min\left\{\mathcal{E}\left(w_t^\top x_{ti}, y_{ti} : i \in \mathbb{N}_{m_t}, t \in \mathbb{N}_n\right) + \gamma\,\Omega(W) : W \in \mathbf{M}_{d,n}\right\}, \tag{20}$$

where $\mathcal{E} : \mathbb{R}^M \times \mathcal{Y}^M \to \mathbb{R}, M = \sum_{t \in \mathbb{N}_n} m_t$. The error term expresses the objective that the regression vector for each task should fit well the data for this particular task. Note, however, that this term need not separate as the sum of functions of the individual task vectors. The choice of the regularizer Ω is important in that it captures certain relationships between the tasks. One common choice is the *trace norm*, which is defined to be the sum of the singular values of a matrix or, equivalently,

$$\Omega(W) = \|W\|_1 := \mathrm{tr}(W^\top W)^{\frac{1}{2}}\,.$$

Regularization with the trace norm learns the tasks as one joint optimization problem and can be seen as a convex relaxation of regularization with the rank (Fazel et al., 2001). It has been shown that for certain configurations of the input data the low rank solution can be recovered using the trace norm approach (Candès and Recht, 2009; Recht et al., 2008). More generally, regardless of the rank of the solution, it has been demonstrated that this approach allows for accurate estimation of related tasks even when there are only *few* data points available for each task.

Thus, it is natural to consider optimization problems of the form (18). In fact, these problems can be seen as instances of problems of the form (15), because the quantity $w_t^\top x_{ti}$ can be written as the inner product between W and a matrix having all its columns equal to zero except for the t-th column being equal to x_{ti}. It is also easy to see that (15) is a richer class since the corresponding constraints are less restrictive.

Despite this fact, by focusing on the class (18) we concentrate on problems of more practical interest and we can obtain representer theorems for a richer class of regularizers, which includes

the trace norm and other useful functions. In contrast, regularization with the functional form (17) is not a satisfactory approach since it ignores matrix structure. In particular, regularization with the Frobenius norm (and a separable error function) corresponds to learning each task *independently*, ignoring relationships among the tasks.

A representer theorem of the form (19) for regularization with the trace norm has been shown in Argyriou et al. (2008a). Related results have also appeared in Abernethy et al. (2009) and Amit et al. (2007). We repeat here the statement and the proof of this theorem, in order to better motivate our proof technique of Section 4.2.

Theorem 12 *If Ω is the trace norm then problem* (18) *(or problem* (20)*) admits a solution $\hat{W}$ of the form* (19)*, for some $c_{si}^{(t)} \in \mathbb{R},\ i \in \mathbb{N}_{m_s}, s,t \in \mathbb{N}_n$.*

Proof Let $\hat{W}$ be a solution of (18) and let $\mathcal{L} := \operatorname{span}\{x_{si} : s \in \mathbb{N}_n, i \in \mathbb{N}_{m_s}\}$. We can decompose the columns of $\hat{W}$ as $\hat{w}_t = \bar{w}_t + p_t,\ \forall t \in \mathbb{N}_n$, where $\bar{w}_t \in \mathcal{L}$ and $p_t \in \mathcal{L}^\perp$. Hence $\hat{W} = \bar{W} + P$, where $\bar{W}$ is the matrix with columns $\bar{w}_t$ and P is the matrix with columns p_t. Moreover we have that $P^\top \bar{W} = 0$. From Lemma 21 in the appendix, we obtain that $\|\hat{W}\|_1 \geq \|\bar{W}\|_1$. We also have that $\hat{w}_t^\top x_{ti} = \bar{w}_t^\top x_{ti}$, for every $i \in \mathbb{N}_{m_t}, t \in \mathbb{N}_n$. Thus, $\bar{W}$ preserves the interpolation constraints (or the value of the error term) while not increasing the value of the regularizer. Hence, it is a solution of the optimization problem and the assertion follows. ■

A simple but important observation about this and related results is that each task vector w_t is a linear combination of the data for *all* the tasks. This contrasts to the representation form (16) obtained by using Frobenius inner product constraints. Interpreting (16) in a multi-task context, by appropriately choosing the X_i as described above, would imply that each w_t is a linear combination of only the data for task t.

Finally, in some applications the following variant, similar to the type (18), has appeared,

$$\min\{\Omega(W) : W \in \mathbf{M}_{d,n},\ w_t^\top x_i = y_{ti},\ \ \forall i \in \mathbb{N}_m, t \in \mathbb{N}_n\}\ . \tag{21}$$

Problems of this type correspond to a special case in multi-task learning applications in which the input data are the same for all the tasks. For instance, this is the case with collaborative filtering or applications in marketing where the same products/entities are rated by all users/consumers (see, for example, Aaker et al., 2004; Evgeniou et al., 2005; Lenk et al., 1996; Srebro et al., 2005, for various approaches to this problem).

4.2 Characterization of Matrix Regularizers

Our objective in this section will be to state and prove a general representer theorem for problems of the form (18) or (21) using a functional form analogous to (12). The key insight used in the proof of Argyriou et al. (2008a) has been that the trace norm is defined in terms of a matrix function that preserves the partial ordering of matrices. That is, it satisfies Lemma 21, which is a matrix analogue of the geometric property (13). To prove our main result (Theorem 15), we shall build on this observation in a way similar to the approach followed in Section 3.

We shall focus on the interpolation problems (18) and (21). First of all, observe that, by definition, problems of the type (18) include those of type (21). Conversely, consider a simple problem of type (18) with two constraints, $W_{11} = 1, W_{22} = 1$. If the set of matrices satisfying these constraints

also satisfied constraints of the form $w_t^\top x_1 = y_{t1}$ for some $x_1 \in \mathbb{R}^d, y_{t1} \in \mathcal{Y}$, then x_1 would have to be the zero vector. Therefore, the class of problems (18) is strictly larger than the class (21).

However, it turns out that with regard to representer theorems of the form (19) there is no distinction between these two classes of problems. In other words, the representer theorem holds for the same regularizers Ω, independently of whether each task has its own specific inputs or the inputs are the same across the tasks. More importantly, we can connect the existence of representer theorems to a geometric property of the regularizer, in a way analogous to property (13) in Section 3. These facts are stated in the following proposition.

Proposition 13 *The following statements are equivalent:*

(a) *Problem* (21) *admits a solution of the form* (19)*, for every data set* $\{(x_i, y_{ti}) : i \in \mathbb{N}_m, t \in \mathbb{N}_n\} \subseteq \mathbb{R}^d \times \mathcal{Y}$ *and every* $m \in \mathbb{N}$*, such that the interpolation constraints are satisfiable.*

(b) *Problem* (18) *admits a solution of the form* (19)*, for every data set* $\{(x_{ti}, y_{ti}) : i \in \mathbb{N}_{m_t}, t \in \mathbb{N}_n\} \subseteq \mathbb{R}^d \times \mathcal{Y}$ *and every* $\{m_t : t \in \mathbb{N}_n\} \subseteq \mathbb{N}$*, such that the interpolation constraints are satisfiable.*

(c) *The function* Ω *satisfies the property*

$$\Omega(W+P) \geq \Omega(W) \qquad \forall\, W, P \in \mathbf{M}_{d,n} \text{ such that } W^\top P = 0\,. \tag{22}$$

Proof We will show that (a) $\Longrightarrow$ (c), (c) $\Longrightarrow$ (b) and (b) $\Longrightarrow$ (a).

[(a) $\Longrightarrow$ (c)] Consider any $W \in \mathbf{M}_{d,n}$. Choose $m = n$ and the input data to be the columns of W. In other words, consider the problem

$$\min\{\Omega(Z) : Z \in \mathbf{M}_{d,n}, Z^\top W = W^\top W\}\,.$$

By hypothesis, there exists a solution $\hat{Z} = WC$ for some $C \in \mathbf{M}_{n,n}$. Since $(\hat{Z} - W)^\top W = 0$, all columns of $\hat{Z} - W$ have to belong to the null space of $W^\top$. But, at the same time, they have to lie in the range of W and hence we obtain that $\hat{Z} = W$. Therefore, we obtain property (22) after the variable change $P = Z - W$.

[(c) $\Longrightarrow$ (b)] Consider arbitrary $x_{ti} \in \mathbb{R}^d, y_{ti} \in \mathcal{Y}, \forall i \in \mathbb{N}_{m_t}, t \in \mathbb{N}_n$, and let $\hat{W}$ be a solution to problem (18). We can decompose the columns of $\hat{W}$ as $\hat{w}_t = \bar{w}_t + p_t$, where $\bar{w}_t \in \mathcal{L} := \text{span}\{x_{si}, i \in \mathbb{N}_{m_s}, s \in \mathbb{N}_n\}$ and $p_t \in \mathcal{L}^\perp$, $\forall t \in \mathbb{N}_n$. By hypothesis $\Omega(\hat{W}) \geq \Omega(\bar{W})$. Since $\hat{W}$ interpolates the data, so does $\bar{W}$ and therefore $\bar{W}$ is a solution to (18).

[(b) $\Longrightarrow$ (a)] Trivial, since any problem of type (21) is also of type (18). ■

We remark in passing that, by a similar proof, property (22) is also equivalent to representer theorems (19) for the class of problems (15).

The above proposition provides us with a criterion for characterizing all regularizers satisfying representer theorems of the form (19), in the context of problems (15), (18) or (21). Our objective will be to obtain a functional form analogous to (12) that describes functions satisfying property (22). This property does not have a simple geometric interpretation, unlike (13) which describes functions with spherical contours. The reason is that matrix products are more difficult to tackle than inner products.

Similar to the Hilbert space setting (12), where we required h to be a nondecreasing real function, the functional description of the regularizer now involves the notion of a *matrix nondecreasing* function.

Definition 14 *We say that the function $h : \mathbf{S}_+^n \to \mathbb{R}$ is nondecreasing in the order of matrices if $h(A) \leq h(B)$ for all $A, B \in \mathbf{S}_+^n$ such that $A \preceq B$.*

Theorem 15 *Let $d, n \in \mathbb{N}$ with $d \geq 2n$. The differentiable function $\Omega : \mathbf{M}_{d,n} \to \mathbb{R}$ satisfies property* (22) *if and only if there exists a matrix nondecreasing function $h : \mathbf{S}_+^n \to \mathbb{R}$ such that*

$$\Omega(W) = h(W^\top W) \qquad \forall\, W \in \mathbf{M}_{d,n}. \tag{23}$$

Proof We first assume that Ω satisfies property (22). From this property it follows that, for all $W, P \in \mathbf{M}_{d,n}$ with $W^\top P = 0$,

$$\langle \nabla\Omega(W), P \rangle = 0. \tag{24}$$

To see this, observe that if the matrix $W^\top P$ is zero then, for all $\varepsilon > 0$, we have that

$$\frac{\Omega(W + \varepsilon P) - \Omega(W)}{\varepsilon} \geq 0.$$

Taking the limit as $\varepsilon \to 0^+$ we obtain that $\langle \nabla\Omega(W), P \rangle \geq 0$. Similarly, choosing $\varepsilon < 0$ we obtain that $\langle \nabla\Omega(W), P \rangle \leq 0$ and equation (24) follows.

Now, consider any matrix $W \in \mathbf{M}_{d,n}$. Let $r = \operatorname{rank}(W)$ and let us write W in a singular value decomposition as follows

$$W = \sum_{i \in \mathbb{N}_r} \sigma_i u_i v_i^\top,$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ are the singular values and $u_i \in \mathbb{R}^d$, $v_i \in \mathbb{R}^n$, $\forall i \in \mathbb{N}_r$, are sets of singular vectors, so that $u_i^\top u_j = v_i^\top v_j = \delta_{ij}$, $\forall i, j \in \mathbb{N}_r$. Also, let $u_{r+1}, \ldots, u_d \in \mathbb{R}^d$ be vectors that together with $u_1, \ldots, u_r$ form an orthonormal basis of $\mathbb{R}^d$. Without loss of generality, let us pick u_1 and consider any *unit* vector z *orthogonal* to the vectors $u_2, \ldots, u_r$. Let $k = d - r + 1$ and $q \in \mathbb{R}^k$ be the unit vector such that

$$z = Rq,$$

where $R = (u_1, u_{r+1}, \ldots, u_d)$. We can complete q by adding $d - r$ columns to its right in order to form an orthogonal matrix $Q \in \mathbf{O}^k$ and, since $d > n$, we may select these columns so that $\det(Q) = 1$. Furthermore, we can write this matrix as $Q = e^D$ with $D \in \mathbf{M}_{k,k}$ a skew-symmetric matrix (see Horn and Johnson, 1991, Example 6.2.15).

We also define the path $Z : [0,1] \to \mathbf{M}_{d,n}$ as

$$Z(\lambda) = \sigma_1 R e^{\lambda D} e_1 v_1^\top + \sum_{i=2}^{r} \sigma_i u_i v_i^\top \qquad \forall \lambda \in [0,1],$$

where e_1 denotes the vector $(1\,0 \ldots 0)^\top \in \mathbb{R}^k$. In other words, we fix the singular values, the right singular vectors and the $r-1$ left singular vectors $u_2, \ldots, u_r$ and only allow the first left singular vector to vary. This path has the properties that $Z(0) = W$ and $Z(1) = \sigma_1 z v_1^\top + \sum_{i=2}^r \sigma_i u_i v_i^\top$.

By construction of the path, it holds that

$$Z'(\lambda) = \sigma_1 R e^{\lambda D} D e_1 v_1^\top$$

and hence

$$Z(\lambda)^\top Z'(\lambda) = \left(\sigma_1 R e^{\lambda D} e_1 v_1^\top\right)^\top \sigma_1 R e^{\lambda D} D e_1 v_1^\top = \sigma_1^2 v_1 e_1^\top D e_1 v_1^\top = 0\,,$$

for every $\lambda \in (0,1)$, because $D_{11} = 0$. Hence, using equation (24), we have that

$$\langle \nabla\Omega(Z(\lambda)), Z'(\lambda)\rangle = 0$$

and, since $\dfrac{d\Omega(Z(\lambda))}{d\lambda} = \langle \nabla\Omega(Z(\lambda)), Z'(\lambda)\rangle$, we conclude that $\Omega(Z(\lambda))$ equals a constant independent of λ. In particular, $\Omega(Z(0)) = \Omega(Z(1))$, that is,

$$\Omega(W) = \Omega\left(\sigma_1 z v_1^\top + \sum_{i=2}^{r} \sigma_i u_i v_i^\top\right).$$

In other words, if we fix the singular values of W, the right singular vectors and all the left singular vectors but one, Ω does not depend on the remaining left singular vector (because the choice of z is independent of u_1).

In fact, this readily implies that Ω does not depend on the left singular vectors at all. Indeed, fix an arbitrary $Y \in \mathbf{M}_{d,n}$ such that $Y^\top Y = I$. Consider the matrix $Y(W^\top W)^{\frac{1}{2}}$, which can be written using the same singular values and right singular vectors as W. That is,

$$Y(W^\top W)^{\frac{1}{2}} = \sum_{i \in \mathbb{N}_r} \sigma_i \tau_i v_i^\top\,,$$

where $\tau_i = Y v_i,\ \forall i \in \mathbb{N}_r$. Now, we select unit vectors $z_1, \ldots, z_r \in \mathbb{R}^d$ as follows:

$$\begin{aligned}
&z_1 = u_1\\
&z_2 \perp z_1, u_3, \ldots, u_r, \tau_1\\
&\quad\vdots\quad\vdots\\
&z_r \perp z_1, \ldots, z_{r-1}, \tau_1, \ldots, \tau_{r-1}\,.
\end{aligned}$$

This construction is possible since $d \geq 2n$. Replacing successively u_i with z_i and then z_i with τ_i, $\forall i \in \mathbb{N}_r$, and applying the invariance property, we obtain that

$$\begin{aligned}
\Omega(W) &= \Omega\left(\sum_{i\in\mathbb{N}_r} \sigma_i u_i v_i^\top\right) \\
&= \Omega\left(\sigma_1 z_1 v_1^\top + \sigma_2 z_2 v_2^\top + \sum_{i=3}^{r} \sigma_i u_i v_i^\top\right) \\
&\quad \vdots \quad \vdots \\
&= \Omega\left(\sum_{i\in\mathbb{N}_r} \sigma_i z_i v_i^\top\right) \\
&= \Omega\left(\sigma_1 \tau_1 v_1^\top + \sum_{i=2}^{r} \sigma_i z_i v_i^\top\right) \\
&\quad \vdots \quad \vdots \\
&= \Omega\left(\sum_{i\in\mathbb{N}_r} \sigma_i \tau_i v_i^\top\right) = \Omega\left(Y(W^\top W)^{\frac{1}{2}}\right).
\end{aligned}$$

Therefore, defining the function $h : \mathbf{S}_+^n \to \mathbb{R}$ as $h(A) = \Omega(YA^{\frac{1}{2}})$, we deduce that $\Omega(W) = h(W^\top W)$.

Finally, we show that h is matrix nondecreasing, that is, $h(A) \leq h(B)$ if $0 \preceq A \preceq B$. For any such A, B and since $d \geq 2n$, we may define $W = \begin{pmatrix} A^{\frac{1}{2}} \\ 0 \\ 0 \end{pmatrix}$, $P = \begin{pmatrix} 0 \\ (B-A)^{\frac{1}{2}} \\ 0 \end{pmatrix} \in \mathbf{M}_{d,n}$. Then $W^\top P = 0$, $A = W^\top W$, $B = (W+P)^\top (W+P)$ and thus, by hypothesis,

$$h(B) = \Omega(W+P) \geq \Omega(W) = h(A).$$

This completes the proof in one direction of the theorem.

To show the converse, assume that $\Omega(W) = h(W^\top W)$, where the function h is matrix nondecreasing. Then for any $W, P \in \mathbf{M}_{d,n}$ with $W^\top P = 0$, we have that $(W+P)^\top (W+P) = W^\top W + P^\top P \succeq W^\top W$ and, so, $\Omega(W+P) \geq \Omega(W)$, as required. ■

As in Section 3, differentiability is not required for sufficiency, which follows from the last lines of the above proof. Moreover, the assumption on d and n is not required either.

Remark 16 *Let the function $\Omega : \mathbf{M}_{d,n} \to \mathbb{R}$ be of the form*

$$\Omega(W) = h(W^\top W) \qquad \forall\, W \in \mathbf{M}_{d,n},$$

for some matrix nondecreasing function $h : \mathbf{S}_+^n \to \mathbb{R}$. Then Ω satisfies property (22).

We conclude this section by providing a necessary and sufficient first-order condition for the function h to be matrix nondecreasing.

Proposition 17 *Let $h : \mathbf{S}_+^n \to \mathbb{R}$ be a differentiable function. The following properties are equivalent:*

(a) h is matrix nondecreasing

(b) the matrix $\nabla h(A) := \left(\frac{\partial h}{\partial a_{ij}} : i, j \in \mathbb{N}_n\right)$ *is positive semidefinite, for every* $A \in \mathbf{S}_+^n$.

Proof If (a) holds, we choose any $x \in \mathbb{R}^n, t > 0, A \in \mathbf{S}_+^n$ and note that

$$\frac{h(A + txx^\top) - h(A)}{t} \geq 0.$$

Letting t go to zero gives that $x^\top \nabla h(A) x \geq 0$.

Conversely, if (b) is true, we have, for every $x \in \mathbb{R}^n, M \in \mathbf{S}_+^n$, that $x^\top \nabla h(M) x = \langle \nabla h(M), xx^\top \rangle \geq 0$ and, so, $\langle \nabla h(M), C \rangle \geq 0$ for all $C \in \mathbf{S}_+^n$. For any $A, B \in \mathbf{S}_+^n$ such that $A \preceq B$, consider the univariate function $g : [0,1] \to \mathbb{R}$, $g(t) = h(A + t(B - A))$. By the chain rule it is easy to verify that g is nondecreasing. Therefore we conclude that $h(A) = g(0) \leq g(1) = h(B)$. ■

4.3 Examples

Using Proposition 13, Theorem 15 and Remark 16, one may easily verify that the representer theorem holds for a variety of regularizers. In particular, functional description (23) subsumes the special case of *monotone spectral functions*.

Definition 18 *Let* $r = \min\{d, n\}$. *A function* $\Omega : \mathbf{M}_{d,n} \to \mathbb{R}$ *is called* monotone spectral *if there exists a function* $h : \mathbb{R}_+^r \to \mathbb{R}$ *such that*

$$\Omega(W) = h(\sigma_1(W), \ldots, \sigma_r(W)) \qquad \forall\, W \in \mathbf{M}_{d,n},$$

where $\sigma_1(W) \geq \cdots \geq \sigma_r(W) \geq 0$ *denote the singular values of matrix* W *and*

$$h(\sigma_1, \ldots, \sigma_r) \leq h(\tau_1, \ldots, \tau_r) \quad \textit{whenever} \;\; \sigma_i \leq \tau_i \textit{ for all } i \in \mathbb{N}_r.$$

Corollary 19 *Assume that the function* $\Omega : \mathbf{M}_{d,n} \to \mathbb{R}$ *is monotone spectral. Then* Ω *satisfies property* (22).

Proof Clearly, $\Omega(W) = h\left(\lambda\left((W^\top W)^{\frac{1}{2}}\right)\right)$, where $\lambda : \mathbf{S}_+^n \to \mathbb{R}_+^r$ maps a matrix to the ordered vector of its r highest eigenvalues. Let $A, B \in \mathbf{S}_+^n$ such that $A \preceq B$. Weyl's monotonicity theorem (Horn and Johnson, 1985, Cor. 4.3.3) states that if $A \preceq B$ then the spectra of A and B are ordered. Thus, $\lambda(A^{\frac{1}{2}}) \leq \lambda(B^{\frac{1}{2}})$ and hence $h(\lambda(A^{\frac{1}{2}})) \leq h(\lambda(B^{\frac{1}{2}}))$. Applying Remark 16, the assertion follows. ■

We note that related results to the above corollary have appeared in Abernethy et al. (2009). They apply to the setting of (15) when the X_i are rank one matrices.

Interesting examples of monotone spectral functions are the *Schatten L_p norms* and *prenorms*,

$$\Omega(W) = \|W\|_p := \|\sigma(W)\|_p,$$

where $p \in [0, +\infty)$ and $\sigma(W)$ denotes the $\min\{d, n\}$-dimensional vector of the singular values of W. For instance, we have already mentioned in Section 4.1 that the representer theorem holds when the regularizer is the trace norm (the L_1 norm of the spectrum). But it also holds for the *rank* of a matrix.

Rank minimization is an NP-hard optimization problem (Vandenberghe and Boyd, 1996), but the representer theorem has some interesting implications. First, that the problem can be reformulated in terms of reproducing kernels and second, that an equivalent problem in as few variables as the total sample size can be obtained.

If we exclude spectral functions, the functions that remain are invariant under *left* multiplication with an orthogonal matrix. Examples of such functions are Schatten norms and prenorms composed with *right* matrix scaling,

$$\Omega(W) = \|WM\|_p, \tag{25}$$

where $M \in \mathbf{M}_{n,n}$. In this case, the corresponding h is the function $S \mapsto \|\sqrt{\lambda(M^\top SM)}\|_p$. To see that this function is matrix nondecreasing, observe that if $A, B \in \mathbf{S}_+^n$ and $A \preceq B$ then $0 \preceq M^\top AM \preceq M^\top BM$ and hence $\lambda(M^\top AM) \leq \lambda(M^\top BM)$ by Weyl's monotonicity theorem (Horn and Johnson, 1985, Cor. 4.3.3). Therefore, $\|\sqrt{\lambda(M^\top AM)}\|_p \leq \|\sqrt{\lambda(M^\top BM)}\|_p$.

For instance, the matrix M above can be used to select a subset of the columns of W. In addition, more complicated structures can be obtained by summation of matrix nondecreasing functions and by taking minima or maxima over sets. For example, we can obtain a regularizer such as

$$\Omega(W) = \min\left\{\sum_{k\in\mathbb{N}_K} \|W(I_k)\|_1 : \{I_1,\ldots,I_K\} \in \mathcal{P}\right\},$$

where $\mathcal{P}$ is the set of partitions of $\mathbb{N}_n$ in K subsets and $W(I_k)$ denotes the submatrix of W formed by just the columns indexed by I_k. This regularizer is an extension of the trace norm ($K = 1$) and can be used for multi-task learning via dimensionality reduction on multiple subspaces (Argyriou et al., 2008b).

Yet another example of a valid regularizer is the one considered in (Evgeniou et al., 2005, Sec. 3.1), which encourages the tasks to be close to each other, namely

$$\Omega(W) = \sum_{t\in\mathbb{N}_n} \left\| w_t - \frac{1}{n}\sum_{s\in\mathbb{N}_n} w_s \right\|^2.$$

This regularizer immediately verifies property (22), and so by Theorem 15 it is a matrix nondecreasing function of $W^\top W$. One can also verify that this regularizer is the square of the form (25) with $p = 2$ and $M = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$, where $\mathbf{1}$ denotes the n-dimensional vector all of whose components are equal to one.

Finally, it is worth noting that the representer theorem does *not* apply to a family of "mixed" matrix norms of the form

$$\Omega(W) = \sum_{i\in\mathbb{N}_d} \|w^i\|_2^p,$$

where w^i denotes the i-th row of W and $p > 1,\ p \neq 2$.

5. Conclusion

We have characterized the classes of vector and matrix regularizers which lead to certain forms of the solution of the associated interpolation problems. In the vector case, we have proved the necessity of a well-known sufficient condition for the "standard representer theorem", which is encountered in many learning and statistical estimation problems. In the matrix case, we have described

a novel class of regularizers which lead to a modified representer theorem. This class, which relies upon the notion of matrix nondecreasing function, includes and extends significantly the vector class. To motivate the need for our study, we have discussed some examples of regularizers which have been recently used in the context of multi-task learning and collaborative filtering.

In the future, it would be valuable to study in more detail special cases of the matrix regularizers which we have encountered, such as those based on orthogonally invariant functions. It would also be interesting to investigate how the presence of additional constraints affects the representer theorem. In particular, we have in mind the possibility that the matrix may be constrained to be in a convex cone, such as the set of positive semidefinite matrices. Finally, we leave to future studies the extension of the ideas presented here to the case in which matrices are replaced by operators between two Hilbert spaces.

Acknowledgments

The work of the first and third authors was supported by EPSRC Grant EP/D052807/1 and by the IST Programme of the European Union, under the PASCAL Network of Excellence IST-2002-506778. The second author is supported by NSF grant DMS 0712827.

Appendix A.

Here we collect some auxiliary results which are used in the above analysis.

The first lemma states a basic property of connectedness through rotations.

Lemma 20 *Let $w, v \in \mathbb{R}^d$ and $d \geq 2$. Then there exists $U \in \mathbf{O}^d$ with determinant 1 such that $v = Uw$ if and only if $\|w\| = \|v\|$.*

Proof If $v = Uw$ we have that $v^\top v = w^\top w$. Conversely, if $\|w\| = \|v\| \neq 0$, we may choose orthonormal vectors $\{x_\ell : \ell \in \mathbb{N}_{d-1}\} \perp w$ and $\{z_\ell : \ell \in \mathbb{N}_{d-1}\} \perp v$ and form the matrices $R = \begin{pmatrix} w & x_1 & \ldots & x_{d-1} \end{pmatrix}$ and $S = \begin{pmatrix} v & z_1 & \ldots & z_{d-1} \end{pmatrix}$. We have that $R^\top R = S^\top S$. We wish to solve the equation $UR = S$. For this purpose we choose $U = SR^{-1}$ and note that $U \in \mathbf{O}^d$ because $U^\top U = (R^{-1})^\top S^T S R^{-1} = (R^{-1})^\top R^\top R R^{-1} = I$. Since $d \geq 2$, in the case that $\det(U) = -1$ we can simply change the sign of one of the x_ℓ or z_ℓ to get $\det(U) = 1$ as required. ■

The second lemma concerns the monotonicity of the trace norm.

Lemma 21 *Let $W, P \in \mathbf{M}_{d,n}$ such that $W^\top P = 0$. Then $\|W + P\|_1 \geq \|W\|_1$.*

Proof Weyl's monotonicity theorem (Horn and Johnson, 1985, Cor. 4.3.3) implies that if $A, B \in \mathbf{S}_+^n$ and $A \succeq B$ then $\lambda(A) \geq \lambda(B)$ and hence $\operatorname{tr} A^{\frac{1}{2}} \geq \operatorname{tr} B^{\frac{1}{2}}$. We apply this fact to the matrices $W^\top W + P^\top P$ and $P^\top P$ to obtain that

$$\|W + P\|_1 = \operatorname{tr}((W+P)^\top (W+P))^{\frac{1}{2}} = \operatorname{tr}(W^\top W + P^\top P)^{\frac{1}{2}} \geq \operatorname{tr}(W^\top W)^{\frac{1}{2}} = \|W\|_1 .$$

■

References

D. A. Aaker, V. Kumar, and G. S. Day. *Marketing Research*. John Wiley & Sons, 2004. 8th edition.

J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering: operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826, 2009.

Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, 2007.

R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, 2006.

A. Argyriou, C. A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, 2007.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008a. Special Issue on Inductive Transfer Learning; Guest Editors: D. L. Silver and K. P. Bennett.

A. Argyriou, A. Maurer, and M. Pontil. An algorithm for transfer learning in a heterogeneous environment. In *Proceedings of the European Conference on Machine Learning*, volume 5211 of *Lecture Notes in Computer Science*, pages 71–85. Springer, 2008b.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 686:337–404, 1950.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM, 1992.

E. J. Candès and B. Recht. Exact matrix completion via convex optimization. In *Foundations of Computational Mathematics*, 2009. To appear.

G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. In *Proceedings of the Twenty-First Annual Conference on Learning Theory*, 2008.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49, 2001.

E. De Vito, L. Rosasco, A. Caponnetto, M. Piana, and A. Verri. Some properties of regularized kernel methods. *Journal of Machine Learning Research*, 5:1363–1390, 2004.

F. Dinuzzo, M. Neve, G. De Nicolao, and U. P. Gianazza. On the representer theorem and equivalent degrees of freedom of SVR. *Journal of Machine Learning Research*, 8:2467–2495, 2007.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings, American Control Conference*, volume 6, pages 4734–4739, 2001.

F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42:80–86, 1970.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

A. J. Izenman. Reduced-rank regression for the multivariate linear model. *Journal of Multivariate Analysis*, 5:248–264, 1975.

G.S. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, 1970.

P. J. Lenk, W. S. DeSarbo, P. E. Green, and M. R. Young. Hierarchical Bayes conjoint analysis: recovery of partworth heterogeneity from reduced experimental designs. *Marketing Science*, 15 (2):173–191, 1996.

A. Maurer. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, 7:117–139, 2006a.

A. Maurer. The Rademacher complexity of linear transformation classes. In *Proceedings of the Nineteenth Annual Conference on Learning Theory*, volume 4005 of *LNAI*, pages 65–78. Springer, 2006b.

C. A. Micchelli and A. Pinkus. Variational problems arising from balancing several error criteria. *Rendiconti di Matematica, Serie VII*, 14:37–86, 1994.

C. A. Micchelli and M. Pontil. A function representation for learning in Banach spaces. In *Proceedings of the Nineteenth Annual Conference on Learning Theory*, volume 3120 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2004.

C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005a.

C. A. Micchelli and M. Pontil. On learning vector–valued functions. *Neural Computation*, 17: 177–204, 2005b.

C. A. Micchelli and T. J. Rivlin. Lectures on optimal recovery. In P. R. Turner, editor, *Lecture Notes in Mathematics*, volume 1129. Springer Verlag, 1985.

R. Raina, A. Y. Ng, and D. Koller. Constructing informative priors using transfer learning. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, 2006.

B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum rank solutions to linear matrix equations via nuclear norm minimization. Preprint, 2008.

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

B. Schölkopf, R. Herbrich, and A.J. Smola. A generalized representer theorem. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, 2001.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005.

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*. V. H. Winston and Sons (distributed by Wiley), 1977. F. John, Translation Editor.

L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

G. Wahba. Multivariate function and operator estimation, based on smoothing splines and reproducing kernels. In *Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity*, volume XII. Addison-Wesley, 1992.

L. Wolf, H. Jhuang, and T. Hazan. Modeling appearances with low-rank SVM. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, 2007.

Z. Xiang and K. P. Bennett. Inductive transfer using kernel multitask latent analysis. In *Inductive Transfer : 10 Years Later, NIPS Workshop*, 2005.

T. Xiong, J. Bi, B. Rao, and V. Cherkassky. Probabilistic joint feature selection for multi-task learning. In *Proceedings of SIAM International Conference on Data Mining*, 2006.

M. Yuan, A. Ekici, Z. Lu, and R. Monteiro. Dimension reduction and coefficient estimation in multivariate linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(3):329–346, 2007.

Journal of Machine Learning Research 10 (2009) 2531-2569 Submitted 3/08; Revised 11/08; Published 11/09

Maximum Entropy Discrimination Markov Networks

Jun Zhu JUNZHU@CS.CMU.EDU
Eric P. Xing EPXING@CS.CMU.EDU
Machine Learning Department
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213

Editor: Michael Collins

Abstract

The standard maximum margin approach for structured prediction lacks a straightforward probabilistic interpretation of the learning scheme and the prediction rule. Therefore its unique advantages such as dual sparseness and kernel tricks cannot be easily conjoined with the merits of a probabilistic model such as Bayesian regularization, model averaging, and ability to model hidden variables. In this paper, we present a new general framework called *maximum entropy discrimination Markov networks* (MaxEnDNet, or simply, MEDN), which integrates these two approaches and combines and extends their merits. Major innovations of this approach include: 1) It extends the conventional max-entropy discrimination learning of classification rules to a new *structural* max-entropy discrimination paradigm of learning a distribution of Markov networks. 2) It generalizes the extant Markov network structured-prediction rule based on a point estimator of model coefficients to an averaging model akin to a Bayesian predictor that integrates over a learned posterior distribution of model coefficients. 3) It admits flexible entropic regularization of the model during learning. By plugging in different prior distributions of the model coefficients, it subsumes the well-known maximum margin Markov networks (M^3N) as a special case, and leads to a model similar to an L_1-regularized M^3N that is simultaneously primal and dual sparse, or other new types of Markov networks. 4) It applies a modular learning algorithm that combines existing variational inference techniques and convex-optimization based M^3N solvers as subroutines. Essentially, MEDN can be understood as a jointly maximum likelihood and maximum margin estimate of Markov network. It represents the first successful attempt to combine maximum entropy learning (a dual form of maximum likelihood learning) with maximum margin learning of Markov network for structured input/output problems; and the basic principle can be generalized to learning arbitrary graphical models, such as the generative Bayesian networks or models with structured hidden variables. We discuss a number of theoretical properties of this approach, and show that empirically it outperforms a wide array of competing methods for structured input/output learning on both synthetic and real OCR and web data extraction data sets.

Keywords: maximum entropy discrimination, structured input/output model, maximum margin Markov network, graphical models, entropic regularization, L_1 regularization

1. Introduction

Inferring structured predictions based on high-dimensional, often multi-modal and hybrid covariates remains a central problem in data mining (e.g., web-info extraction), machine intelligence (e.g., machine translation), and scientific discovery (e.g., genome annotation). Several recent approaches to this problem are based on learning discriminative graphical models defined on composite features

that explicitly exploit the structured dependencies among input elements and structured interpretational outputs. Major instances of such models include the conditional random fields (CRFs) (Lafferty et al., 2001), Markov networks (MNs) (Taskar et al., 2003), and other specialized graphical models (Altun et al., 2003). Various paradigms for training such models based on different loss functions have been explored, including the maximum conditional likelihood learning (Lafferty et al., 2001) and the maximum margin learning (Altun et al., 2003; Taskar et al., 2003; Tsochantaridis et al., 2004), with remarkable success.

The likelihood-based models for structured predictions are usually based on a joint distribution of both input and output variables (Rabiner, 1989) or a conditional distribution of the output given the input (Lafferty et al., 2001). Therefore this paradigm offers a flexible probabilistic framework that can naturally facilitate: hidden variables that capture latent semantics such as a generative hierarchy (Quattoni et al., 2004; Zhu et al., 2008a); Bayesian regularization that imposes desirable biases such as sparseness (Lee et al., 2006; Wainwright et al., 2006; Andrew and Gao, 2007); and Bayesian prediction based on combining predictions across all values of model parameters (i.e., model averaging), which can reduce the risk of overfitting. On the other hand, the margin-based structured prediction models leverage the maximum margin principle and convex optimization formulation underlying the support vector machines, and concentrate directly on the input-output mapping (Taskar et al., 2003; Altun et al., 2003; Tsochantaridis et al., 2004). In principle, this approach can lead to a robust decision boundary due to the dual sparseness (i.e., depending on only a few support vectors) and global optimality of the learned model. However, although arguably a more desirable paradigm for training highly discriminative structured prediction models in a number of application contexts, the lack of a straightforward probabilistic interpretation of the maximum-margin models makes them unable to offer the same flexibilities of likelihood-based models discussed above.

For example, for domains with complex feature space, it is often desirable to pursue a "sparse" representation of the model that leaves out irrelevant features. In likelihood-based estimation, sparse model fitting has been extensively studied. A commonly used strategy is to add an L_1-penalty to the likelihood function, which can also be viewed as a MAP estimation under a Laplace prior. However, little progress has been made so far on learning sparse MNs or log-linear models in general based on the maximum margin principle. While sparsity has been pursued in maximum margin learning of certain discriminative models such as SVM that are "unstructured" (i.e., with a univariate output), by using L_1-regularization (Bennett and Mangasarian, 1992) or by adding a cardinality constraint (Chan et al., 2007), generalization of these techniques to structured output space turns out to be non-trivial, as we discuss later in this paper. There is also very little theoretical analysis on the performance guarantee of margin-based models under direct L_1-regularization. Our empirical results as shown in this paper suggest that an L_1-regularized maximum margin Markov network, even when estimable, can be sensitive to the magnitude of the regularization coefficient. Discarding the features that are not completely irrelevant can potentially hurt generalization ability. Another example, it is well known that presence of hidden variables in MNs can cause significant difficulty for maximum margin learning. Indeed, semi-supervised or unsupervised learning of structured maximum margin model remains an open problem of which progress was only made in a few special cases, with usually computationally very expensive algorithms (Xu et al., 2006; Altun et al., 2006; Brefeld and Scheffer, 2006).

In this paper, we propose a general theory of maximum entropy discrimination Markov networks (MaxEnDNet, or simply MEDN) for structured input/output learning and prediction. This formalism offers a formal paradigm for integrating both generative and discriminative principles and

the Bayesian regularization techniques for learning structured prediction models. It integrates the spirit of maximum margin learning from SVM, the design of discriminative structured prediction model in maximum margin Markov networks (M^3N), and the ideas of entropic regularization and model averaging in maximum entropy discrimination methods (Jaakkola et al., 1999). Essentially, MaxEnDNet can be understood as a jointly maximum likelihood and maximum margin estimate of Markov networks. It allows one to learn a *distribution* of structured prediction models that offers a wide range of important advantages over conventional models such as M^3N, including more robust prediction due to an averaging prediction-function based on the learned distribution of models, Bayesian-style regularization that can lead to a model that is simultaneous primal and dual sparse, and allowance of hidden variables and semi-supervised learning based on partially labeled data.

While the formalism of MaxEnDNet is extremely general, our main focus and contributions of this paper will be concentrated on the following results. We will formally define the MaxEnDNet as solving a generalized entropy optimization problem subject to expected margin constraints on the structured predictions, and under an arbitrary prior of feature coefficients; and we derive a general form of the solution to this problem. An interesting insight immediately following this general form is that, a trivial assumption on the prior distribution of the coefficients, that is, a standard normal, reduces the linear MaxEnDNet to the standard M^3N, as shown in Theorem 3. This understanding opens the way to use different priors for MaxEnDNet to achieve more interesting regularization effects. We show that, by using a Laplace prior for the feature coefficients, the resulting Laplace MaxEnDNet (LapMEDN) is effectively an M^3N that is not only dual sparse (i.e., defined by a few support vectors), but also primal sparse (i.e., shrinkage on coefficients corresponding to irrelevant features). We develop a novel variational learning method for the LapMEDN, which leverages on the hierarchical/scale-mixture representation of the Laplace prior (Andrews and Mallows, 1974; Figueiredo, 2003) and the reducibility of MaxEnDNet to M^3N, and combines the variational Bayesian technique with existing convex optimization algorithms developed for M^3N (Taskar et al., 2003; Bartlett et al., 2004; Ratliff et al., 2007). We also provide a formal analysis of the generalization error of the MaxEnDNet, and prove a PAC-Bayes bound on the prediction error by MaxEnDNet. We performed a thorough comparison of the Laplace MaxEnDNet with competing methods, including M^3N (i.e., the Gaussian MaxEnDNet), L_1-regularized M^3N (Zhu et al., 2009b), CRFs, L_1-regularized CRFs, and L_2-regularized CRFs, on both synthetic and real structured input/output data. The Laplace MaxEnDNet exhibits mostly superior, and sometimes comparable performance in all scenarios been tested.

As demonstrated in our recent work (Zhu et al., 2008c, 2009a), MaxEnDNet is not limited to fully observable MNs, but can readily facilitate jointly maximum entropy and maximum margin learning of partially observed structured I/O models, and directed graphical models such as the supervised latent Dirichlet allocation (LDA). Due to space limit, we leave these instantiations and generalizations to future papers.

The rest of the paper is structured as follows. In the next section, we review the basic structured prediction formalism and set the stage for our model. Section 3 presents the general theory of maximum entropy discrimination Markov networks and some basic theoretical results, followed by two instantiations of the general MaxEnDNet, the Gaussian MaxEnDNet and the Laplace MaxEnDNet. Section 4 offers a detailed discussion of the primal and dual sparsity property of Laplace MaxEnDNet. Section 5 presents a novel iterative learning algorithm based on variational approximation and convex optimization. In Section 6, we briefly discuss the generalization bound of MaxEnDNet.

Then, we show empirical results on both synthetic and real OCR and web data extraction data sets in Section 7. Section 8 discusses some related work and Section 9 concludes this paper.

2. Preliminaries

In structured prediction problems such as natural language parsing, image annotation, or DNA decoding, one aims to learn a function $h : \mathcal{X} \to \mathcal{Y}$ that maps a structured input $\mathbf{x} \in \mathcal{X}$, e.g., a sentence or an image, to a structured output $\mathbf{y} \in \mathcal{Y}$, e.g., a sentence parsing or a scene annotation, where, unlike a standard classification problem, $\mathbf{y}$ is a multivariate prediction consisting of multiple labeling elements. Let L denote the cardinality of the output, and m_l where $l = 1, \ldots, L$ denote the arity of each element, then $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_L$ with $\mathcal{Y}_l = \{a_1, \ldots, a_{m_l}\}$ represents a combinatorial space of structured interpretations of the multi-facet objects in the inputs. For example, $\mathcal{Y}$ could correspond to the space of all possible instantiations of the parse trees of a sentence, or the space of all possible ways of labeling entities over some segmentation of an image. The prediction $\mathbf{y} \equiv (y_1, \ldots, y_L)$ is *structured* because each individual label $y_l \in \mathcal{Y}_l$ within $\mathbf{y}$ must be determined in the context of other labels $y_{l' \neq l}$, rather than independently as in classification, in order to arrive at a globally satisfactory and consistent prediction.

Let $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ represent a discriminant function over the input-output pairs from which one can define the predictive function, and let $\mathcal{H}$ denote the space of all possible F. A common choice of F is a linear model, $F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = g(\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}))$, where $\mathbf{f} = [f_1 \ldots f_K]^\top$ is a K-dimensional column vector of the feature functions $f_k : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, and $\mathbf{w} = [w_1 \ldots w_K]^\top$ is the corresponding vector of the weights of the feature functions. Typically, a structured prediction model chooses an optimal estimate $\mathbf{w}^\star$ by minimizing some loss function $J(\mathbf{w})$, and defines a predictive function in terms of an optimization problem that maximizes $F(\,\cdot\,; \mathbf{w}^\star)$ over the response variable $\mathbf{y}$ given an input $\mathbf{x}$:

$$h_0(\mathbf{x}; \mathbf{w}^\star) = \arg \max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} F(\mathbf{x}, \mathbf{y}; \mathbf{w}^\star), \tag{1}$$

where $\mathcal{Y}(\mathbf{x}) \subseteq \mathcal{Y}$ is the feasible subset of structured labels for the input $\mathbf{x}$. Here, we assume that $\mathcal{Y}(\mathbf{x})$ is finite for any $\mathbf{x}$.

Depending on the specific choice of $g(\cdot)$ (e.g., linear, or log linear), and the loss function $J(\mathbf{w})$ (e.g., likelihood, or margin-based loss) for estimating the parameter $\mathbf{w}^\star$, incarnations of the general structured prediction formalism described above can be seen in classical generative models such as the HMM (Rabiner, 1989) where $g(\cdot)$ can be an exponential family distribution function and $J(\mathbf{w})$ is the (negative) joint likelihood of the input and its labeling; and in recent discriminative models such as CRFs (Lafferty et al., 2001), where $g(\cdot)$ is a Boltzmann machine and $J(\mathbf{w})$ is the (negative) conditional likelihood of the structured labeling given input; and the M^{3}N (Taskar et al., 2003), where $g(\cdot)$ is an identity function and $J(\mathbf{w})$ is a loss defined on the margin between the true labeling and any other feasible labeling in $\mathcal{Y}(\mathbf{x})$. Our approach toward a more general discriminative training is based on a maximum entropy principle that allows an elegant combination of the discriminative maximum margin learning with the generative Bayesian regularization and hierarchical modeling, and we consider the more general problem of finding a distribution of $F(\,\cdot\,; \mathbf{w})$ over $\mathcal{H}$ that enables a convex combination of discriminant functions for robust structured prediction.

Before delving into the exposition of the proposed approach, we end this section with a brief recapitulation of the basic M^{3}N, upon which the proposed approach is built. Under a max-margin

framework, given a set of fully observed training data $\mathcal{D} = \{\langle \mathbf{x}^i, \mathbf{y}^i \rangle\}_{i=1}^N$, we obtain a point estimate of the weight vector $\mathbf{w}$ by solving the following max-margin problem P0 (Taskar et al., 2003):

$$\begin{aligned} \text{P0 (M}^3\text{N)}: \quad & \min_{\mathbf{w},\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \\ \text{s.t. } \forall i, \forall \mathbf{y} \neq \mathbf{y}^i: \quad & \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i,\ \xi_i \geq 0\ , \end{aligned}$$

where $\Delta\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) - \mathbf{f}(\mathbf{x}^i, \mathbf{y})$ and $\Delta F_i(\mathbf{y}; \mathbf{w}) = \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y})$ is the "margin" between the true label $\mathbf{y}^i$ and a prediction $\mathbf{y}$, $\Delta\ell_i(\mathbf{y})$ is a labeling loss with respect to $\mathbf{y}^i$, and ξ_i represents a slack variable that absorbs errors in the training data. Various forms of the labeling loss have been proposed in the literature (Tsochantaridis et al., 2004). In this paper, we adopt the *hamming loss* used by Taskar et al. (2003): $\Delta\ell_i(\mathbf{y}) = \sum_{j=1}^{L} \mathbb{I}(y_j \neq y_j^i)$, where $\mathbb{I}(\cdot)$ is an indicator function that equals to one if the argument is true and zero otherwise. The problem P0 is not directly solvable by using a standard constrained optimization toolbox because the feasible space for $\mathbf{w}$,

$$\mathcal{F}_0 = \left\{ \mathbf{w} : \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i;\ \forall i, \forall \mathbf{y} \neq \mathbf{y}^i \right\},$$

is defined by $O(N|\mathcal{Y}|)$ number of constraints, and $\mathcal{Y}$ is exponential to the size of the input $\mathbf{x}$. Exploring sparse dependencies among individual labels y_l in $\mathbf{y}$, as reflected in the specific design of the feature functions (e.g., based on pair-wise labeling potentials in a pair-wise Markov network), and the convex duality of the objective, efficient optimization algorithms based on cutting-plane (Tsochantaridis et al., 2004) or message-passing (Taskar et al., 2003) have been proposed to obtain an approximate optimum solution to P0. As described shortly, these algorithms can be directly employed as subroutines in solving our proposed model.

3. Maximum Entropy Discrimination Markov Networks

Instead of learning a point estimator of $\mathbf{w}$ as in M^{3}N, in this paper, we take a Bayesian-style approach and learn a distribution $p(\mathbf{w})$, in a max-margin manner. For prediction, we employ a convex combination of all possible models $F(\,\cdot\,; \mathbf{w}) \in \mathcal{H}$ based on $p(\mathbf{w})$, that is:

$$h_1(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \int p(\mathbf{w}) F(\mathbf{x}, \mathbf{y}; \mathbf{w})\, \mathrm{d}\mathbf{w}\,. \tag{2}$$

Now, the open question underlying this averaging prediction rule is how we can devise an appropriate loss function and constraints over $p(\mathbf{w})$, in a similar spirit as the margin-based scheme over $\mathbf{w}$ in P0, that lead to an optimum estimate of $p(\mathbf{w})$. In the sequel, we present *Maximum Entropy Discrimination Markov Networks*, a novel framework that facilitates the estimation of a Bayesian-style regularized *distribution* of M^{3}Ns defined by $p(\mathbf{w})$. As we show below, this new Bayesian-style max-margin learning formalism offers several advantages such as simultaneous primal and dual sparsity, PAC-Bayesian generalization guarantee, and estimation robustness. Note that the MaxEnDNet is different from the traditional Bayesian methods for discriminative structured prediction such as the Bayesian CRFs (Qi et al., 2005), where the likelihood function is well defined. Here, our approach is of a "Bayesian-style" because it learns and uses a "posterior" distribution of all predictive models instead of choosing one model according to some criterion, but the learning algorithm is not

based on the Bayes theorem, but a maximum entropy principle that biases towards a distribution that makes less additional assumptions over a given prior over the predictive models. We emphasize that this "posterior" is different from, and should not be confused with, the conventional Bayesian posterior defined according to the Bayes rule.

It is well-known that exponential family distributions can be expressed variationally as the solution to a maximum entropy estimation subject to moment constraints, and the maximum entropy estimation of parameters can be understood as a dual to the maximum likelihood estimation of the parameters of exponential family distributions. Thus our combination of the maximum entropy principle with the maximum margin principle to be presented in the sequel offers an elegant way of achieving jointly maximum margin and maximum likelihood effects on learning structured input/output Markov networks, and in fact, general exponential family graphical models.

3.1 Structured Maximum Entropy Discrimination

Given a training set $\mathcal{D}$ of structured input-output pairs, analogous to the feasible space $\mathcal{F}_0$ for the weight vector $\mathbf{w}$ in a standard M^{3}N (c.f., problem P0), we define the feasible subspace $\mathcal{F}_1$ for the weight distribution $p(\mathbf{w})$ by a set of *expected* margin constraints:

$$\mathcal{F}_1 = \Big\{ p(\mathbf{w}) : \int p(\mathbf{w})[\Delta F_i(\mathbf{y};\mathbf{w}) - \Delta\ell_i(\mathbf{y})]\,\mathrm{d}\mathbf{w} \geq -\xi_i,\ \forall i, \forall \mathbf{y} \neq \mathbf{y}^i \Big\}.$$

We learn the optimum $p(\mathbf{w})$ from $\mathcal{F}_1$ based on a *structured maximum entropy discrimination principle* generalized from the maximum entropy discrimination (Jaakkola et al., 1999). Under this principle, the optimum $p(\mathbf{w})$ corresponds to the distribution that minimizes its relative entropy with respect to some chosen prior p_0, as measured by the Kullback-Leibler divergence between p and p_0: $KL(p||p_0) = \langle \log(p/p_0) \rangle_p$, where $\langle\cdot\rangle_p$ denotes the expectations with respect to p. If p_0 is uniform, then minimizing this KL-divergence is equivalent to maximizing the entropy $H(p) = -\langle \log p \rangle_p$. A natural information theoretic interpretation of this formulation is that we favor a distribution over the discriminant function class $\mathcal{H}$ that bears minimum assumptions among all feasible distributions in $\mathcal{F}_1$. The p_0 is a regularizer that introduces an appropriate bias, if necessary.

To accommodate non-separable cases in the discriminative prediction problem, instead of minimizing the usual KL, we optimize the *generalized entropy* (Dudík et al., 2007; Lebanon and Lafferty, 2001), or a regularized KL-divergence, $KL(p(\mathbf{w})||p_0(\mathbf{w})) + U(\xi)$, where $U(\xi)$ is a closed proper convex function over the slack variables. This term can be understood as an additional "potential" in the maximum entropy principle. Putting everything together, we can now state a general formalism based on the following maximum entropy discrimination Markov network framework:

Definition 1 (Maximum Entropy Discrimination Markov Networks) *Given training data $\mathcal{D} = \{\langle \mathbf{x}^i, \mathbf{y}^i \rangle\}_{i=1}^N$, a chosen form of discriminant function $F(\mathbf{x},\mathbf{y};\mathbf{w})$, a loss function $\Delta\ell(\mathbf{y})$, and an ensuing feasible subspace $\mathcal{F}_1$ (defined above) for parameter distribution $p(\mathbf{w})$, the MaxEnDNet model that leads to a prediction function of the form of Equation (2) is defined by the following generalized relative entropy minimization with respect to a parameter prior $p_0(\mathbf{w})$:*

$$\text{P1 (MaxEnDNet)}: \quad \min_{p(\mathbf{w}),\xi} KL(p(\mathbf{w})||p_0(\mathbf{w})) + U(\xi) \quad \text{s.t. } p(\mathbf{w}) \in \mathcal{F}_1,\ \xi_i \geq 0, \forall i.$$

The P1 defined above is a variational optimization problem over $p(\mathbf{w})$ in a subspace of valid parameter distributions. Since both the KL and the function U in P1 are convex, and the constraints in $\mathcal{F}_1$ are linear, P1 is a convex program. In addition, the expectations $\langle F(\mathbf{x},\mathbf{y};\mathbf{w})\rangle_{p(\mathbf{w})}$ are required to be bounded in order for F to be a meaningful model. Thus, the problem P1 satisfies the *Slater's condition*[1] (Boyd and Vandenberghe, 2004, chap. 5), which together with the convexity make P1 enjoy nice properties, such as strong duality and the existence of solutions. The problem P1 can be solved via applying the calculus of variations to the Lagrangian to obtain a variational extremum, followed by a dual transformation of P1. We state the main results below as a theorem, followed by a brief proof that lends many insights into the solution to P1 which we will explore in subsequent analysis.

Theorem 2 (Solution to MaxEnDNet) *The variational optimization problem P1 underlying the MaxEnDNet gives rise to the following optimum distribution of Markov network parameters* $\mathbf{w}$*:*

$$p(\mathbf{w}) = \frac{1}{Z(\alpha)} p_0(\mathbf{w}) \exp\Big\{ \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})[\Delta F_i(\mathbf{y};\mathbf{w}) - \Delta\ell_i(\mathbf{y})] \Big\}, \tag{3}$$

where $Z(\alpha)$ *is a normalization factor and the Lagrange multipliers* $\alpha_i(\mathbf{y})$ *(corresponding to the constraints in* $\mathcal{F}_1$*) can be obtained by solving the dual problem of P1:*

$$\begin{aligned} \text{D1}: \quad & \max_{\alpha} \; -\log Z(\alpha) - U^\star(\alpha) \\ & \text{s.t. } \alpha_i(\mathbf{y}) \geq 0, \; \forall i, \; \forall \mathbf{y} \neq \mathbf{y}^i \end{aligned}$$

where $U^\star(\cdot)$ *is the conjugate of the slack function* $U(\cdot)$*, that is,* $U^\star(\alpha) = \sup_\xi \big(\sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\xi_i - U(\xi)\big)$.

Proof (*sketch*) Since the problem P1 is a convex program and satisfies the Slater's condition, we can form a Lagrangian function, whose saddle point gives the optimal solution of P1 and D1, by introducing a non-negative dual variable $\alpha_i(\mathbf{y})$ for each constraint in $\mathcal{F}_1$ and another non-negative dual variable c for the normalization constraint $\int p(\mathbf{w})\,\mathrm{d}\mathbf{w} = 1$. Details are deferred to Appendix B.1. ■

Since the problem P1 is a convex program and satisfies the Slater's condition, the saddle point of the Lagrangian function is the KKT point of P1. From the KKT conditions (Boyd and Vandenberghe, 2004, Chap. 5), it can be shown that the above solution enjoys *dual sparsity*, that is, only a few Lagrange multipliers will be non-zero, which correspond to the active constraints whose equality holds, analogous to the support vectors in SVM. Thus MaxEnDNet enjoys a similar generalization property as the M^{3}N and SVM due to the the small "effective size" of the margin constraints. But it is important to realize that this does not mean that the learned model is "primal-sparse", that is, only a few elements in the weight vector $\mathbf{w}$ are non-zero. We will return to this point in Section 4.

For a closed proper convex function $\phi(\mu)$, its conjugate is defined as $\phi^\star(\nu) = \sup_\mu[\nu^\top\mu - \phi(\mu)]$. In the problem D1, by convex duality (Boyd and Vandenberghe, 2004), the log normalizer $\log Z(\alpha)$ can be shown to be the conjugate of the KL-divergence. If the slack function is $U(\xi) = C\|\xi\| =$

1. Since $\langle F(\mathbf{x},\mathbf{y};\mathbf{w})\rangle_{p(\mathbf{w})}$ are bounded and $\xi_i \geq 0$, there always exists a ξ, which is large enough to make the pair $(p(\mathbf{w}),\xi)$ satisfy the Slater's condition.

$C\sum_i \xi_i$, it is easy to show that $U^\star(\alpha) = \mathbb{I}_\infty(\sum_{\mathbf{y}} \alpha_i(\mathbf{y}) \leq C,\ \forall i)$, where $\mathbb{I}_\infty(\cdot)$ is a function that equals to zero when its argument holds true and infinity otherwise. Here, the inequality corresponds to the trivial solution $\xi = 0$, that is, the training data are perfectly separable. Ignoring this inequality does not affect the solution since the special case $\xi = 0$ is still included. Thus, the Lagrange multipliers $\alpha_i(\mathbf{y})$ in the dual problem D1 comply with the set of constraints that $\sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = C,\ \forall i$. Another example is $U(\xi) = KL(p(\xi)||p_0(\xi))$ by introducing uncertainty on the slack variables (Jaakkola et al., 1999). In this case, expectations with respect to $p(\xi)$ are taken on both sides of all the constraints in $\mathcal{F}_1$. Take the duality, and the dual function of U is another log normalizer. More details were provided by Jaakkola et al. (1999). Some other U functions and their dual functions are studied by Lebanon and Lafferty (2001) and Dudík et al. (2007).

Unlike most extant structured discriminative models including the highly successful M^{3}N, which rely on a point estimator of the parameters, the MaxEnDNet model derived above gives an optimum parameter distribution, which is used to make prediction via the rule (2). Indeed, as we will show shortly, the MaxEnDNet is strictly more general than the M^{3}N and subsumes the later as a special case. But more importantly, the MaxEnDNet in its full generality offers a number of important advantages while retaining all the merits of the M^{3}N. **First**, MaxEnDNet admits a prior that can be designed to introduce useful regularization effects, such as a primal sparsity bias. **Second**, the MaxEnDNet prediction is based on model averaging and therefore enjoys a desirable smoothing effect, with a uniform convergence bound on generalization error. **Third**, MaxEnDNet offers a principled way to incorporate *hidden* generative models underlying the structured predictions, but allows the predictive model to be discriminatively trained based on partially labeled data. In the sequel, we analyze the first two points in detail; exploration of the third point is beyond the scope of this paper, and can be found in Zhu et al. (2008c), where a *partially observed* MaxEnDNet (PoMEN) is developed, which combines (possibly latent) generative model and discriminative training for structured prediction.

3.2 Gaussian MaxEnDNet

As Equation (3) suggests, different choices of the parameter prior can lead to different MaxEnDNet models for predictive parameter distribution. In this subsection and the following one, we explore a few common choices, e.g., Gaussian and Laplace priors.

We first show that, when the parameter prior is set to be a standard normal, MaxEnDNet leads to a predictor that is identical to that of the M^{3}N. This somewhat surprising reduction offers an important insight for understanding the property of MaxEnDNet. Indeed this result should not be totally unexpected given the striking isomorphisms of the opt-problem P1, the feasible space $\mathcal{F}_1$, and the predictive function h_1 underlying a MaxEnDNet, to their counterparts P0, $\mathcal{F}_0$, and h_0, respectively, underlying an M^{3}N. The following theorem makes our claim explicit.

Theorem 3 (Gaussian MaxEnDNet: Reduction of MEDN to M^{3}N) *Assuming* $F(\mathbf{x},\mathbf{y};\mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y})$, $U(\xi) = C\sum_i \xi_i$, *and* $p_0(\mathbf{w}) = \mathcal{N}(\mathbf{w}|0,I)$, *where I denotes an identity matrix, then the posterior distribution is* $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mu,I)$, *where* $\mu = \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})$, *and the Lagrange multipliers* $\alpha_i(\mathbf{y})$ *in* $p(\mathbf{w})$ *are obtained by solving the following dual problem, which is isomorphic to the dual form of the M^{3}N:*

$$\max_{\alpha} \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y}) - \frac{1}{2}\|\sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})\|^2$$

$$\text{s.t.} \quad \sum_{\mathbf{y} \neq \mathbf{y}^i} \alpha_i(\mathbf{y}) = C;\ \alpha_i(\mathbf{y}) \geq 0,\ \forall i,\ \forall \mathbf{y} \neq \mathbf{y}^i,$$

where $\Delta \mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) - \mathbf{f}(\mathbf{x}^i, \mathbf{y})$ *as in P0. When applied to* h_1*,* $p(\mathbf{w})$ *leads to a predictive function that is identical to* $h_0(\mathbf{x}; \mathbf{w})$ *given by Equation (1).*

Proof See Appendix B.2 for details. ■

The above theorem is stated in the duality form. We can also show the following equivalence in the primal form.

Corollary 4 *Under the same assumptions as in Theorem 3, the mean* μ *of the posterior distribution* $p(\mathbf{w})$ *under a Gaussian MaxEnDNet is obtained by solving the following primal problem:*

$$\begin{aligned} &\min_{\mu, \xi} \ \frac{1}{2}\mu^\top \mu + C \sum_{i=1}^{N} \xi_i \\ &\text{s.t.} \ \ \mu^\top \Delta \mathbf{f}_i(\mathbf{y}) \geq \Delta \ell_i(\mathbf{y}) - \xi_i;\ \xi_i \geq 0, \ \ \forall i,\ \forall \mathbf{y} \neq \mathbf{y}^i. \end{aligned}$$

Proof See Appendix B.3 for details. ■

Theorem 3 and Corollary 4 both show that in the supervised learning setting, the M^{3}N is a special case of MaxEnDNet when the slack function is linear and the parameter prior is a standard normal. As we shall see later, this connection renders many existing techniques for solving the M^{3}N directly applicable for solving the MaxEnDNet.

3.3 Laplace MaxEnDNet

Recent trends in pursuing "sparse" graphical models has led to the emergence of regularized version of CRFs (Andrew and Gao, 2007) and Markov networks (Lee et al., 2006; Wainwright et al., 2006). Interestingly, while such extensions have been successfully implemented by several authors in maximum likelihood learning of various sparse graphical models, they have not yet been fully explored or evaluated in the context of maximum margin learning, although some existing methods can be extended to achieve sparse max-margin estimators, as explained below.

One possible way to learn a sparse M^{3}N is to adopt the strategy of L_1-SVM (Bennett and Mangasarian, 1992; Zhu et al., 2004) and directly use an L_1 instead of the L_2-norm of $\mathbf{w}$ in the loss function (see appendix A for a detailed description of this formulation and the duality derivation). However, the primal problem of an L_1-regularized M^{3}N is not directly solvable using a standard optimization toolbox by re-formulating it as an LP problem due to the exponential number of constraints; solving the dual problem, which now has only a polynomial number of constraints as in the dual of M^{3}N, is also non-trivial due to the complicated form of the constraints. The constraint generation methods (Tsochantaridis et al., 2004) are possible. However, although such methods have been shown to be efficient for solving the QP problem in the standard M^{3}N, our preliminary empirical results show that such a scheme with an LP solver for the L_1-regularized M^{3}N can be extremely expensive for a non-trivial real data set. Another type of possible solvers are based on a projection to L_1-ball (Duchi et al., 2008), such as the gradient descent (Ratliff et al., 2007) and the dual extragradient (Taskar et al., 2006) methods.

The MaxEnDNet interpretation of the M^{3}N offers an alternative strategy that resembles Bayesian regularization (Tipping, 2001; Kaban, 2007) in maximum likelihood estimation, where shrinkage effects can be introduced by appropriate priors over the model parameters. As Theorem 3 reveals, an M^{3}N corresponds to a Gaussian MaxEnDNet that admits a standard normal prior for the weight vector $\mathbf{w}$. According to the standard Bayesian regularization theory, to achieve a sparse estimate of a model, in the posterior distribution of the feature weights, the weights of irrelevant features should peak around zero with very small variances. However, the isotropy of the variances in all dimensions of the feature space under a standard normal prior makes it infeasible for the resulting M^{3}N to adjust the variances in different dimensions to fit a sparse model. Alternatively, now we employ a Laplace prior for $\mathbf{w}$ to learn a Laplace MaxEnDNet. We show in the sequel that, the parameter posterior $p(\mathbf{w})$ under a Laplace MaxEnDNet has a shrinkage effect on small weights, which is similar to directly applying an L_1-regularizer on an M^{3}N. Although exact learning of a Laplace MaxEnDNet is also intractable, we show that this model can be efficiently approximated by a variational inference procedure based on existing methods.

The Laplace prior of $\mathbf{w}$ is expressed as $p_0(\mathbf{w}) = \prod_{k=1}^{K} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|w_k|} = (\frac{\sqrt{\lambda}}{2})^K e^{-\sqrt{\lambda}\|\mathbf{w}\|}$. This density function is heavy tailed and peaked at zero; thus, it encodes a prior belief that the distribution of $\mathbf{w}$ is strongly peaked around zero. Another nice property of the Laplace density is that it is log-concave, or the negative logarithm is convex, which can be exploited to obtain a convex estimation problem analogous to LASSO (Tibshirani, 1996).

Theorem 5 (Laplace MaxEnDNet: a sparse M^{3}N) *Assuming* $F(\mathbf{x},\mathbf{y};\mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y})$, $U(\xi) = C\sum_i \xi_i$, *and* $p_0(\mathbf{w}) = \prod_{k=1}^{K} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|w_k|} = (\frac{\sqrt{\lambda}}{2})^K e^{-\sqrt{\lambda}\|\mathbf{w}\|}$, *then the Lagrange multipliers* $\alpha_i(\mathbf{y})$ *in* $p(\mathbf{w})$ *(as defined in Theorem 2) are obtained by solving the following dual problem:*

$$\max_{\alpha} \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y}) - \sum_{k=1}^{K} \log\frac{\lambda}{\lambda-\eta_k^2}$$
$$\text{s.t.} \sum_{\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y}) = C;\ \alpha_i(\mathbf{y}) \geq 0,\ \forall i,\ \forall \mathbf{y} \neq \mathbf{y}^i.$$

where $\eta_k = \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i^k(\mathbf{y})$, *and* $\Delta\mathbf{f}_i^k(\mathbf{y}) = f_k(\mathbf{x}^i,\mathbf{y}^i) - f_k(\mathbf{x}^i,\mathbf{y})$ *represents the kth component of* $\Delta\mathbf{f}_i(\mathbf{y})$. *Furthermore, constraints* $\eta_k^2 < \lambda,\ \forall k$, *must be satisfied*.

Since several intermediate results from the proof of this Theorem will be used in subsequent presentations, we provide the complete proof below. Our proof is based on a hierarchical representation of the Laplace prior. As noted by Andrews and Mallows (1974), the Laplace distribution $p(w) = \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|w|}$ is equivalent to a two-layer hierarchical Gaussian-exponential model, where w follows a zero-mean Gaussian distribution $p(w|\tau) = \mathcal{N}(w|0,\tau)$ and the variance τ admits an exponential hyper-prior density,

$$p(\tau|\lambda) = \frac{\lambda}{2}\exp\{-\frac{\lambda}{2}\tau\},\ \text{ for } \tau \geq 0.$$

This alternative form straightforwardly leads to the following new representation of our multivariate Laplace prior for the parameter vector $\mathbf{w}$ in MaxEnDNet:

$$p_0(\mathbf{w}) = \prod_{k=1}^{K} p_0(w_k) = \prod_{k=1}^{K} \int p(w_k|\tau_k)p(\tau_k|\lambda)\,d\tau_k = \int p(\mathbf{w}|\tau)p(\tau|\lambda)\,d\tau, \tag{4}$$

where $p(\mathbf{w}|\tau) = \prod_{k=1}^{K} p(w_k|\tau_k)$ and $p(\tau|\lambda) = \prod_{k=1}^{K} p(\tau_k|\lambda)$ represent multivariate Gaussian and exponential, respectively, and $d\tau \triangleq d\tau_1 \cdots d\tau_K$.

Proof (*of Theorem 5*) Substitute the hierarchical representation of the Laplace prior (Equation 4) into $p(\mathbf{w})$ in Theorem 2, and we get the normalization factor $Z(\alpha)$ as follows,

$$\begin{aligned}
Z(\alpha) &= \int\int p(\mathbf{w}|\tau)p(\tau|\lambda)\,d\tau \cdot \exp\{\mathbf{w}^\top\eta - \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y})\}\,d\mathbf{w} \\
&= \int p(\tau|\lambda)\int p(\mathbf{w}|\tau)\cdot\exp\{\mathbf{w}^\top\eta - \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y})\}\,d\mathbf{w}\,d\tau \\
&= \int p(\tau|\lambda)\int \mathcal{N}(\mathbf{w}|0,A)\exp\{\mathbf{w}^\top\eta - \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y})\}\,d\mathbf{w}\,d\tau \\
&= \int p(\tau|\lambda)\exp\{\frac{1}{2}\eta^\top A\eta - \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y})\}d\tau \\
&= \exp\{-\sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y})\}\prod_{k=1}^{K}\int\frac{\lambda}{2}\exp(-\frac{\lambda}{2}\tau_k)\exp(\frac{1}{2}\eta_k^2\tau_k)d\tau_k \\
&= \exp\{-\sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y})\}\prod_{k=1}^{K}\frac{\lambda}{\lambda-\eta_k^2}, \qquad (5)
\end{aligned}$$

where $A = \mathrm{diag}(\tau_k)$ is a diagonal matrix and η is a column vector with η_k defined as in Theorem 5. The last equality is due to the moment generating function of an exponential distribution. The constraint $\eta_k^2 < \lambda,\ \forall k$ is needed in this derivation to avoid the integration going infinity. Substituting the normalization factor derived above into the general dual problem D1 in Theorem 2, and using the same argument of the convex conjugate of $U(\xi) = C\sum_i \xi_i$ as in Theorem 3, we arrive at the dual problem in Theorem 5. ■

It can be shown that the dual objective function of Laplace MaxEnDNet in Theorem 5 is concave.[2] But since each η_k depends on all the dual variables α and η_k^2 appears within a logarithm, the optimization problem underlying Laplace MaxEnDNet would be very difficult to solve. The SMO (Taskar et al., 2003) and the exponentiated gradient methods (Bartlett et al., 2004) developed for the QP dual problem of M^{3}N cannot be easily applied here. Thus, we will turn to a variational approximation method, as shown in Section 5. For completeness, we end this section with a corollary similar to the Corollary 4, which states the primal optimization problem underlying the MaxEnDNet with a Laplace prior. As we shall see, the primal optimization problem in this case is complicated and provides another perspective of the hardness of solving the Laplace MaxEnDNet.

Corollary 6 *Under the same assumptions as in Theorem 5, the mean μ of the posterior distribution $p(\mathbf{w})$ under a Laplace MaxEnDNet is obtained by solving the following primal problem:*

$$\begin{aligned}
&\min_{\mu,\xi}\ \sqrt{\lambda}\sum_{k=1}^{K}\Big(\sqrt{\mu_k^2+\frac{1}{\lambda}} - \frac{1}{\sqrt{\lambda}}\log\frac{\sqrt{\lambda\mu_k^2+1}+1}{2}\Big) + C\sum_{i=1}^{N}\xi_i \\
&\text{s.t.}\ \ \mu^\top\Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i;\ \xi_i \geq 0,\ \ \forall i,\ \forall\mathbf{y}\neq\mathbf{y}^i.
\end{aligned}$$

2. η_k^2 is convex over α because it is the composition of $f(x) = x^2$ with an affine mapping. So, $\lambda - \eta_k^2$ is concave and $\log(\lambda - \eta_k^2)$ is also concave due to the composition rule (Boyd and Vandenberghe, 2004).

Proof The proof requires the result of Corollary 7. We defer it to Appendix B.4. ■

Since the "norm"[3]

$$\sum_{k=1}^{K}\Big(\sqrt{\mu_k^2+\frac{1}{\lambda}}-\frac{1}{\sqrt{\lambda}}\log\frac{\sqrt{\lambda\mu_k^2+1}+1}{2}\Big)\triangleq\|\mu\|_{KL}$$

corresponds to the KL-divergence between $p(\mathbf{w})$ and $p_0(\mathbf{w})$ under a Laplace MaxEnDNet, we will refer to it as a *KL-norm* and denote it by $\|\cdot\|_{KL}$ in the sequel. This KL-norm is different from the L_2-norm as used in M^{3}N, but is closely related to the L_1-norm, which encourages a sparse estimator. In the following section, we provide a detailed analysis of the sparsity of Laplace MaxEnDNet resulted from the regularization effect from this norm.

4. Entropic Regularization and Sparse M^{3}N

Comparing to the structured prediction law h_0 due to an M^{3}N, which enjoys dual sparsity (i.e., few support vectors), the h_1 defined by a Laplace MaxEnDNet is not only dual-sparse, but also primal sparse; that is, features that are insignificant will experience strong shrinkage on their corresponding weight w_k.

The primal sparsity of h_1 achieved by the Laplace MaxEnDNet is due to a shrinkage effect resulting from the *Laplacian entropic regularization*. In this section, we take a close look at this regularization effect, in comparison with other common regularizers, such as the L_2-norm in M^{3}N (which is equivalent to the Gaussian MaxEnDNet), and the L_1-norm that at least in principle could be directly applied to M^{3}N. Since our main interest here is the sparsity of the structured prediction law h_1, we examine the posterior mean under $p(\mathbf{w})$ via exact integration. It can be shown that under a Laplace MaxEnDNet, $p(\mathbf{w})$ exhibits the following posterior shrinkage effect.

Corollary 7 (Entropic Shrinkage) *The posterior mean of the Laplace MaxEnDNet has the following form:*

$$\langle w_k\rangle_p=\frac{2\eta_k}{\lambda-\eta_k^2},\ \forall 1\le k\le K, \tag{6}$$

where $\eta_k=\sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})(f_k(\mathbf{x}^i,\mathbf{y}^i)-f_k(\mathbf{x}^i,\mathbf{y}))$ *and* $\eta_k^2<\lambda,\ \forall k$.

Proof Using the integration result in Equation (5), we can get:

$$\frac{\partial\log Z}{\partial\alpha_i(\mathbf{y})}=v^\top\Delta\mathbf{f}_i(\mathbf{y})-\Delta\ell_i(\mathbf{y}), \tag{7}$$

where v is a column vector and $v_k=\frac{2\eta_k}{\lambda-\eta_k^2},\ \forall 1\le k\le K$. An alternative way to compute the derivatives is using the definition of Z: $Z=\int p_0(\mathbf{w})\cdot\exp\{\mathbf{w}^\top\eta-\sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y})\}\,\mathrm{d}\mathbf{w}$. We can get:

$$\frac{\partial\log Z}{\partial\alpha_i(\mathbf{y})}=\langle\mathbf{w}\rangle_p^\top\Delta\mathbf{f}_i(\mathbf{y})-\Delta\ell_i(\mathbf{y}). \tag{8}$$

3. This is not exactly a norm because the positive scalability does not hold. But the KL-norm is non-negative due to the non-negativity of KL-divergence. In fact, by using the inequality $e^x\ge 1+x$, we can show that each component $(\sqrt{\mu_k^2+\frac{1}{\lambda}}-\frac{1}{\sqrt{\lambda}}\log\frac{\sqrt{\lambda\mu_k^2+1}+1}{2})$ is monotonically increasing with respect to μ_k^2 and $\|\mu\|_{KL}\ge K/\sqrt{\lambda}$, where the equality holds only when $\mu=0$. Thus, $\|\mu\|_{KL}$ penalizes large weights. For convenient comparison with the popular L_2 and L_1 norms, we call it a KL-norm.

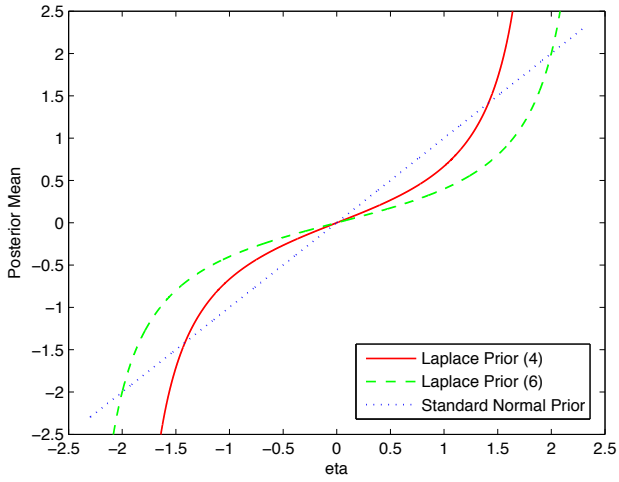


Figure 1: Posterior means with different priors against their corresponding $\eta = \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})$. Note that the η for different priors are generally different because of the different dual parameters.

Comparing Equations (7) and (8), we get $\langle \mathbf{w} \rangle_p = v$, that is, $\langle w_k \rangle_p = \frac{2\eta_k}{\lambda - \eta_k^2}$, $\forall 1 \leq k \leq K$. The constraints $\eta_k^2 < \lambda$, $\forall k$ are required to get a finite normalization factor as shown in Equation (5). ■

Here η is isomorphic to an unregularized estimate of the feature weight vector which directly comes from a linear combination of support vectors (and therefore not sparsified). A plot of the relationship between $\langle w_k \rangle_p$ under a Laplace MaxEnDNet and the corresponding η_k revealed by Corollary 7 is shown in Figure 1 (for example, the red curve), from which we can see that, the smaller the η_k is, the more shrinkage toward zero is imposed on $\langle w_k \rangle_p$.

This entropic shrinkage effect on $\mathbf{w}$ is not present in the standard M^{3}N, and the Gaussian MaxEnDNet. Recall that by definition, the vector $\eta \triangleq \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})$ is determined by the dual parameters $\alpha_i(\mathbf{y})$ obtained by solving a model-specific dual problem. When the $\alpha_i(\mathbf{y})$'s are obtained by solving the dual of the standard M^{3}N, it can be shown that the optimum point solution of the parameters $\mathbf{w}^\star = \eta$. When the $\alpha_i(\mathbf{y})$'s are obtained from the dual of the Gaussian MaxEnDNet, Theorem 3 shows that the posterior mean of the parameters $\langle \mathbf{w} \rangle_{p_{\text{Gaussian}}} = \eta$. (As we have already pointed out, since these two dual problems are isomorphic, the $\alpha_i(\mathbf{y})$'s for M^{3}N and Gaussian MaxEnDNet are identical, hence the resulting η's are the same.) In both cases, there is no shrinkage along any particular dimension of the parameter vector $\mathbf{w}$ or of the mean vector of $p(\mathbf{w})$. Therefore, although both M^{3}N and Gaussian MaxEnDNet enjoy the dual sparsity, because the KKT conditions imply that most of the dual parameters $\alpha_i(\mathbf{y})$'s are zero, $\mathbf{w}^\star$ and $\langle \mathbf{w} \rangle_{p_{\text{Gaussian}}}$ are not primal sparse. From Equation (6), we can conclude that the Laplace MaxEnDNet is also dual sparse, because its mean $\langle \mathbf{w} \rangle_{p_{\text{Laplace}}}$ can be uniquely determined by η. But the shrinkage effect on different components of the $\langle \mathbf{w} \rangle_{p_{\text{Laplace}}}$ vector causes $\langle \mathbf{w} \rangle_{p_{\text{Laplace}}}$ to be also primal sparse.

A comparison of the posterior mean estimates of $\mathbf{w}$ under MaxEnDNet with three different priors versus their associated η is shown in Figure 1. The three priors in question are, a standard normal, a Laplace with $\lambda = 4$, and a Laplace with $\lambda = 6$. It can be seen that, under the entropic

regularization with a Laplace prior, the $\langle \mathbf{w} \rangle_p$ gets shrunk toward zero when η is small. The larger the λ value is, the greater the shrinkage effect. For a fixed λ, the shape of the shrinkage curve (i.e., the $\langle \mathbf{w} \rangle_p - \eta$ curve) is smoothly nonlinear, but no component is explicitly discarded, that is, no weight is set explicitly to zero. In contrast, for the Gaussian MaxEnDNet, which is equivalent to the standard M^{3}N, there is no such a shrinkage effect.

Corollary 6 offers another perspective of how the Laplace MaxEnDNet relates to the L_1-norm M^{3}N, which yields a sparse estimator. Note that as λ goes to infinity, the KL-norm $\|\mu\|_{KL}$ approaches $\|\mu\|_1$, that is, the L_1-norm.[4] This means that the MaxEnDNet with a Laplace prior will be (nearly) the same as the L_1-M^{3}N if the regularization constant λ is large enough.

A more explicit illustration of the entropic regularization under a Laplace MaxEnDNet, comparing to the conventional L_1 and L_2 regularization over an M^{3}N, can be seen in Figure 2, where the feasible regions due to the three different norms used in the regularizer are plotted in a two dimensional space. Specifically, it shows (1) L_2-norm: $w_1^2 + w_2^2 \leq 1$; (2) L_1-norm: $|w_1| + |w_2| \leq 1$; and (3) KL-norm:[5] $\sqrt{w_1^2 + 1/\lambda} + \sqrt{w_2^2 + 1/\lambda} - (1/\sqrt{\lambda}) \log(\sqrt{\lambda w_1^2 + 1}/2 + 1/2) - (1/\sqrt{\lambda}) \log(\sqrt{\lambda w_1^2 + 1}/2 + 1/2) \leq b$, where b is a parameter to make the boundary pass the $(0,1)$ point for easy comparison with the L_2 and L_1 curves. It is easy to show that b equals to $\sqrt{1/\lambda} + \sqrt{1 + 1/\lambda} - (1/\sqrt{\lambda}) \log(\sqrt{\lambda + 1}/2 + 1/2)$. It can be seen that the L_1-norm boundary has sharp turning points when it passes the axises, whereas the L_2 and KL-norm boundaries turn smoothly at those points. This is the intuitive explanation of why the L_1-norm directly gives sparse estimators, whereas the L_2-norm and KL-norm due to a Laplace prior do not. But as shown in Figure 2(b), when the λ gets larger and larger, the KL-norm boundary moves closer and closer to the L_1-norm boundary. When $\lambda \to \infty$, $\sqrt{w_1^2 + 1/\lambda} + \sqrt{w_2^2 + 1/\lambda} - (1/\sqrt{\lambda}) \log(\sqrt{\lambda w_1^2 + 1}/2 + 1/2) - (1/\sqrt{\lambda}) \log(\sqrt{\lambda w_1^2 + 1}/2 + 1/2) \to |w_1| + |w_2|$ and $b \to 1$, which yields exactly the L_1-norm in the two dimensional space. Thus, under the linear model assumption of the discriminant functions $F(\,\cdot\,; \mathbf{w})$, our framework can be seen as a smooth relaxation of the L_1-M^{3}N.

5. Variational Learning of Laplace MaxEnDNet

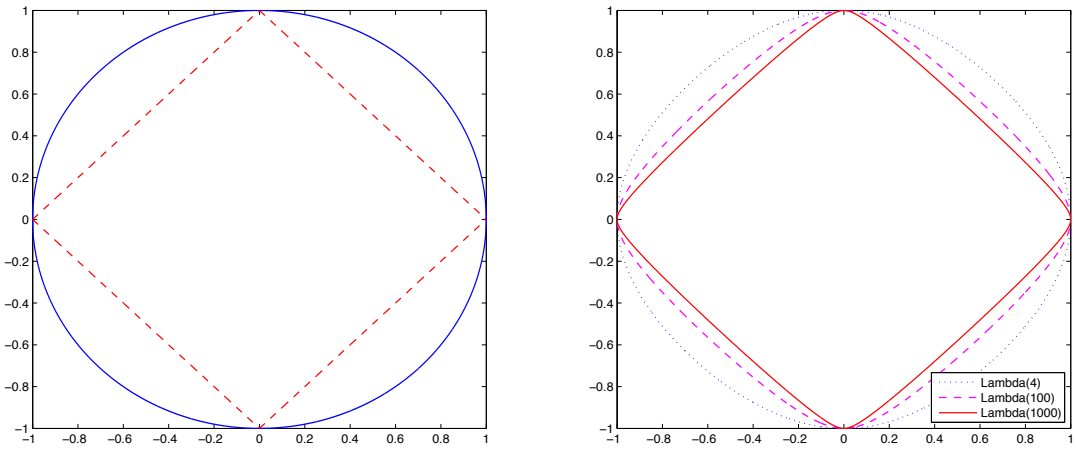
Although Theorem 2 seems to offer a general closed-form solution to $p(\mathbf{w})$ under an arbitrary prior $p_0(\mathbf{w})$, in practice the Lagrange multipliers $\alpha_i(\mathbf{y})$ in $p(\mathbf{w})$ can be very hard to estimate from the dual problem D1 except for a few special choices of $p_0(\mathbf{w})$, such as a normal as shown in Theorem 3, which can be easily generalized to any normal prior. When $p_0(\mathbf{w})$ is a Laplace prior, as we have shown in Theorem 5 and Corollary 6, the corresponding dual problem or primal problem involves a complex objective function that is difficult to optimize. Here, we present a variational method for an approximate learning of the Laplace MaxEnDNet.

Our approach is built on the hierarchical interpretation of the Laplace prior as shown in Equation (4). Replacing the $p_0(\mathbf{w})$ in Problem P1 with Equation (4), and applying the Jensen's inequality, we get an upper bound of the KL-divergence:

$$KL(p||p_0) \qquad = -H(p) - \langle \log \int p(\mathbf{w}|\tau) p(\tau|\lambda)\, \mathrm{d}\tau \rangle_p$$

4. As $\lambda \to \infty$, the logarithm terms in $\|\mu\|_{KL}$ disappear because of the fact that $\frac{\log x}{x} \to 0$ when $x \to \infty$.

5. The curves are drawn with a symbolic computational package to solve an equation of the form: $2x - \log x = a$, where x is the variable to be solved and a is a constant.



(a) (b)

Figure 2: (a) L_2-norm (solid line) and L_1-norm (dashed line); (b) KL-norm with different Laplace priors.

$$\begin{aligned} &\leq -H(p) - \langle \int q(\tau) \log \frac{p(\mathbf{w}|\tau)p(\tau|\lambda)}{q(\tau)} \, \mathrm{d}\tau \rangle_p \\ &\triangleq \mathcal{L}(p(\mathbf{w}), q(\tau)), \end{aligned}$$

where $q(\tau)$ is a variational distribution used to approximate $p(\tau|\lambda)$. The upper bound is in fact a KL-divergence: $\mathcal{L}(p(\mathbf{w}), q(\tau)) = KL(p(\mathbf{w})q(\tau)||p(\mathbf{w}|\tau)p(\tau|\lambda))$. Thus, $\mathcal{L}$ is convex over $p(\mathbf{w})$, and $q(\tau)$, respectively, but not necessarily joint convex over $(p(\mathbf{w}), q(\tau))$.

Substituting this upper bound for the KL-divergence in P1, we now solve the following Variational MaxEnDNet problem,

$$\text{P1}' \text{ (vMEDN)}: \qquad \min_{p(\mathbf{w}) \in \mathcal{F}_1; q(\tau); \xi} \mathcal{L}(p(\mathbf{w}), q(\tau)) + U(\xi).$$

P1$'$ can be solved with an iterative minimization algorithm alternating between optimizing over $(p(\mathbf{w}), \xi)$ and $q(\tau)$, as outlined in Algorithm 1, and detailed below.

Step 1: Keep $q(\tau)$ fixed, optimize P1$'$ with respect to $(p(\mathbf{w}), \xi)$. Using the same procedure as in solving P1, we get the posterior distribution $p(\mathbf{w})$ as follows,

$$\begin{aligned} p(\mathbf{w}) \quad &\propto \exp\{\int q(\tau) \log p(\mathbf{w}|\tau) \, \mathrm{d}\tau - b\} \cdot \exp\{\mathbf{w}^\top \eta - \sum_{i, \mathbf{y} \neq \mathbf{y}^i} \alpha_i(\mathbf{y}) \Delta \ell_i(\mathbf{y})\} \\ &\propto \exp\{-\frac{1}{2} \mathbf{w}^\top \langle A^{-1} \rangle_q \mathbf{w} - b + \mathbf{w}^\top \eta - \sum_{i, \mathbf{y} \neq \mathbf{y}^i} \alpha_i(\mathbf{y}) \Delta \ell_i(\mathbf{y})\} \\ &= \mathcal{N}(\mathbf{w}|\mu, \Sigma), \end{aligned}$$

Algorithm 1 Variational MaxEnDNet

Input: data $\mathcal{D} = \{\langle \mathbf{x}^i, \mathbf{y}^i \rangle\}_{i=1}^N$, constants C and λ, iteration number T
Output: posterior mean $\langle \mathbf{w} \rangle_p^T$
Initialize $\langle \mathbf{w} \rangle_p^1 \leftarrow 0, \Sigma^1 \leftarrow I$
for $t = 1$ **to** $T-1$ **do**
  Step 1: solve (9) or (10) for $\langle \mathbf{w} \rangle_p^{t+1} = \Sigma^t \eta$; update $\langle \mathbf{w}\mathbf{w}^\top \rangle_p^{t+1} \leftarrow \Sigma^t + \langle \mathbf{w} \rangle_p^{t+1} (\langle \mathbf{w} \rangle_p^{t+1})^\top$.
  Step 2: use (11) to update $\Sigma^{t+1} \leftarrow \text{diag}(\sqrt{\frac{\langle w_k^2 \rangle_p^{t+1}}{\lambda}})$.
end for

where $\eta = \sum_{i,\mathbf{y} \neq \mathbf{y}^i} \alpha_i(\mathbf{y}) \Delta \mathbf{f}_i(\mathbf{y})$, $A = \text{diag}(\tau_k)$, and $b = KL(q(\tau)||p(\tau|\lambda))$ is a constant. The posterior mean and variance are $\langle \mathbf{w} \rangle_p = \mu = \Sigma\eta$ and $\Sigma = (\langle A^{-1} \rangle_q)^{-1} = \langle \mathbf{w}\mathbf{w}^\top \rangle_p - \langle \mathbf{w} \rangle_p \langle \mathbf{w} \rangle_p^\top$, respectively. Note that this posterior distribution is also a normal distribution. Analogous to the proof of Theorem 3, we can derive that the dual parameters α are estimated by solving the following dual problem:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i,\mathbf{y} \neq \mathbf{y}^i} \alpha_i(\mathbf{y}) \Delta \ell_i(\mathbf{y}) - \frac{1}{2} \eta^\top \Sigma \eta \\ \text{s.t.} \quad & \sum_{\mathbf{y} \neq \mathbf{y}^i} \alpha_i(\mathbf{y}) = C;\ \alpha_i(\mathbf{y}) \geq 0,\ \forall i,\ \forall \mathbf{y} \neq \mathbf{y}^i. \end{aligned} \tag{9}$$

This dual problem is now a standard quadratic program symbolically identical to the dual of an M^3N, and can be directly solved using existing algorithms developed for M^3N, such as the SMO (Taskar et al., 2003) and the exponentiated gradient (Bartlett et al., 2004) methods. Alternatively, we can solve the following primal problem:

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{1}{2} \mathbf{w}^\top \Sigma^{-1} \mathbf{w} + C \sum_{i=1}^N \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y}) \geq \Delta \ell_i(\mathbf{y}) - \xi_i;\ \xi_i \geq 0,\ \ \forall i,\ \forall \mathbf{y} \neq \mathbf{y}^i. \end{aligned} \tag{10}$$

Based on the proof of Corollary 4, it is easy to show that the solution of the problem (10) leads to the posterior mean of $\mathbf{w}$ under $p(\mathbf{w})$, which will be used to do prediction by h_1. The primal problem can be solved with the subgradient (Ratliff et al., 2007), cutting-plane (Tsochantaridis et al., 2004), or extragradient (Taskar et al., 2006) method.

Step 2: Keep $p(\mathbf{w})$ fixed, optimize P1′ with respect to $q(\tau)$. Taking the derivative of $\mathcal{L}$ with respect to $q(\tau)$ and set it to zero, we get:

$$q(\tau) \quad \propto p(\tau|\lambda) \exp\{\langle \log p(\mathbf{w}|\tau) \rangle_p\}.$$

Since both $p(\mathbf{w}|\tau)$ and $p(\tau|\lambda)$ can be written as a product of univariate Gaussian and univariate exponential distributions, respectively, over each dimension, $q(\tau)$ also factorizes over each dimension: $q(\tau) = \prod_{k=1}^K q(\tau_k)$, where each $q(\tau_k)$ can be expressed as:

$$\begin{aligned} \forall k: \quad q(\tau_k) \quad & \propto p(\tau_k|\lambda) \exp\{\langle \log p(w_k|\tau_k) \rangle_p\} \\ & \propto \mathcal{N}(\sqrt{\langle w_k^2 \rangle_p}|0,\tau_k) \exp(-\frac{1}{2}\lambda\tau_k). \end{aligned}$$

The same distribution has been derived by Kaban (2007), and similar to the hierarchical representation of a Laplace distribution we can get the normalization factor: $\int \mathcal{N}(\sqrt{\langle w_k^2\rangle_p}|0,\tau_k)\cdot \frac{\lambda}{2}\exp(-\frac{1}{2}\lambda\tau_k)\,d\tau_k = \frac{\sqrt{\lambda}}{2}\exp(-\sqrt{\lambda\langle w_k^2\rangle_p})$. Also, we can calculate the expectations $\langle\tau_k^{-1}\rangle_q$ which are required in calculating $\langle A^{-1}\rangle_q$ as follows,

$$\langle\frac{1}{\tau_k}\rangle_q = \int \frac{1}{\tau_k} q(\tau_k)\,d\tau_k = \sqrt{\frac{\lambda}{\langle w_k^2\rangle_p}}. \tag{11}$$

We iterate between the above two steps until convergence. Due to the convexity (not joint convexity) of the upper bound, the algorithm is guaranteed to converge to a local optimum. Then, we apply the posterior distribution $p(\mathbf{w})$, which is in the form of a normal distribution, to make prediction using the averaging prediction law in Equation (2). Due to the shrinkage effect of the Laplacian entropic regularization discussed in Section 4, for irrelevant features, the variances should converge to zeros and thus lead to a sparse estimation of $\mathbf{w}$. To summarize, the intuition behind this iterative minimization algorithm is as follows. First, we use a Gaussian distribution to approximate the Laplace distribution and thus get a QP problem that is analogous to that of the standard M^{3}N; then, in the second step we update the covariance matrix in the QP problem with an exponential hyper-prior on the variance.

6. Generalization Bound

The PAC-Bayes theory for averaging classifiers (McAllester, 1999; Langford et al., 2001) provides a theoretical motivation to learn an averaging model for classification. In this section, we extend the classic PAC-Bayes theory on binary classifiers to MaxEnDNet, and analyze the generalization performance of the structured prediction rule h_1 in Equation (2). In order to prove an error bound for h_1, the following mild assumption on the boundedness of discriminant function $F(\,\cdot\,;\mathbf{w})$ is necessary, that is, there exists a positive constant c, such that,

$$\forall \mathbf{w}, \quad F(\,\cdot\,;\mathbf{w}) \in \mathcal{H}: \quad \mathcal{X}\times\mathcal{Y} \to [-c,c].$$

Recall that the averaging structured prediction function under the MaxEnDNet is defined as $h(\mathbf{x},\mathbf{y}) = \langle F(\mathbf{x},\mathbf{y};\mathbf{w})\rangle_{p(\mathbf{w})}$. Let's define the predictive margin of an instance $(\mathbf{x},\mathbf{y})$ under a function h as $M(h,\mathbf{x},\mathbf{y}) = h(\mathbf{x},\mathbf{y}) - \max_{\mathbf{y}'\neq\mathbf{y}} h(\mathbf{x},\mathbf{y}')$. Clearly, h makes a wrong prediction on $(\mathbf{x},\mathbf{y})$ only if $M(h,\mathbf{x},\mathbf{y}) \leq 0$. Let Q denote a distribution over $\mathcal{X}\times\mathcal{Y}$, and let $\mathcal{D}$ represent a sample of N instances randomly drawn from Q. With these definitions, we have the following structured version of PAC-Bayes theorem.

Theorem 8 (PAC-Bayes Bound of MaxEnDNet) *Let p_0 be any continuous probability distribution over $\mathcal{H}$ and let $\delta\in(0,1)$. If $F(\,\cdot\,;\mathbf{w})\in\mathcal{H}$ is bounded by $\pm c$ as above, then with probability at least $1-\delta$, for a random sample $\mathcal{D}$ of N instances from Q, for every distribution p over $\mathcal{H}$, and for all margin thresholds $\gamma>0$:*

$$\Pr_Q(M(h,\mathbf{x},\mathbf{y})\leq 0) \leq \Pr_{\mathcal{D}}(M(h,\mathbf{x},\mathbf{y})\leq\gamma) + O\Big(\sqrt{\frac{\gamma^{-2}KL(p||p_0)\ln(N|\mathcal{Y}|)+\ln N+\ln\delta^{-1}}{N}}\Big),$$

where $\Pr_Q(.)$ *and* $\Pr_{\mathcal{D}}(.)$ *represent the probabilities of events over the true distribution* Q*, and over the empirical distribution of* $\mathcal{D}$*, respectively.*

The proof of Theorem 8 follows the same spirit of the proof of the original PAC-Bayes bound, but with a number of technical extensions dealing with structured outputs and margins. See appendix B.5 for the details.

Recently, McAllester (2007) presents a *stochastic* max-margin structured prediction model, which is different from the averaging predictor under the MaxEnDNet model, by defining/designing a "posterior" distribution from which a model is sampled to make prediction, and achieves a PAC-Bayes bound which holds for arbitrary models sampled from the particular posterior distribution. Langford and Shawe-Taylor (2003) show an interesting connection between the PAC-Bayes bounds for averaging classifiers and stochastic classifiers, again by designing a posterior distribution. But our posterior distribution is solved with MaxEnDNet and is generally different from those designed by McAllester (2007) and Langford and Shawe-Taylor (2003).

7. Experiments

In this section, we present empirical evaluations of the proposed Laplace MaxEnDNet (LapMEDN) on both synthetic and real data sets. We compare LapMEDN with M^3N (i.e., the Gaussian MaxEnDNet), L_1-regularized M^3N (L_1-M^3N), CRFs, L_1-regularized CRFs (L_1-CRFs), and L_2-regularized CRFs (L_2-CRFs). We use the quasi-Newton method (Liu and Nocedal, 1989) and its variant (Andrew and Gao, 2007) to solve the optimization problem of CRFs, L_1-CRFs, and L_2-CRFs. For M^3N and LapMEDN, we use the exponentiated gradient method (Bartlett et al., 2004) to solve the dual QP problem; and we also use the sub-gradient method (Ratliff et al., 2007) to solve the corresponding primal problem. To the best of our knowledge, no formal description, implementation, and evaluation of the L_1-M^3N exist in the literature, therefore for comparison purpose we had to develop this model and algorithm anew. Details of our work along this line was reported in Zhu et al. (2009b), which is beyond the scope of this paper. But briefly, for our experiments on synthetic data, we implemented the constraint generating method (Tsochantaridis et al., 2004) which uses MOSEK to solve an equivalent LP re-formulation of L_1-M^3N. However, this approach is extremely slow on larger problems; therefore on real data we instead applied the sub-gradient method (Ratliff et al., 2007) with a projection to an L_1-ball (Duchi et al., 2008) to solve the larger L_1-M^3N based on the equivalent re-formulation with an L_1-norm constraint (i.e., the second formulation in Appendix A).

7.1 Evaluation on Synthetic Data

We first evaluate all the competing models on synthetic data where the true structured predictions are known. Here, we consider sequence data, that is, each input $\mathbf{x}$ is a sequence $(x_1, \ldots, x_L)$, and each component x_l is a d-dimensional vector of input features. The synthetic data are generated from pre-specified conditional random field models with either i.i.d. instantiations of the input features (i.e., elements in the d-dimensional feature vectors) or correlated (i.e., structured) instantiations of the input features, from which samples of the structured output $\mathbf{y}$, that is, a sequence $(y_1, \ldots, y_L)$, can be drawn from the conditional distribution $p(\mathbf{y}|\mathbf{x})$ defined by the CRF based on a Gibbs sampler.

7.1.1 I.I.D. INPUT FEATURES

The first experiment is conducted on synthetic sequence data with 100 i.i.d. input features (i.e., $d = 100$). We generate three types of data sets with 10, 30, and 50 relevant input features, respectively. For each type, we randomly generate 10 linear-chain CRFs with 8 binary labeling states (i.e., $L = 8$ and $\mathcal{Y}_l = \{0,1\}$). The feature functions include: a real valued state-feature function over a one dimensional input feature and a class label; and 4 (2×2) binary transition feature functions capturing pairwise label dependencies. For each model we generate a data set of 1000 samples. For each sample, we first *independently* draw the 100 input features from a standard normal distribution, and then apply a Gibbs sampler (based on the conditional distribution of the generated CRFs) to assign a labeling sequence with 5000 iterations.

For each data set, we randomly draw a subset as training data and use the rest for testing. The sizes of training set are 30, 50, 80, 100, and 150. The QP problem in M^{3}N and the first step of LapMEDN is solved with the exponentiated gradient method (Bartlett et al., 2004). In all the following experiments, the regularization constants of L_1-CRFs and L_2-CRFs are chosen from $\{0.01, 0.1, 1, 4, 9, 16\}$ by a 5-fold cross-validation during the training. For the LapMEDN, we use the same method to choose λ from 20 roughly evenly spaced values between 1 and 268. For each setting, a performance score is computed from the average over 10 random samples of data sets.

The results are shown in Figure 3. All the results of the LapMEDN are achieved with 3 iterations of the variational learning algorithm. From the results, we can see that under different settings LapMEDN consistently outperforms M^{3}N and performs comparably with L_1-CRFs and L_1-M^{3}N, both of which encourage a sparse estimate; and both the L_1-CRFs and L_2-CRFs outperform the un-regularized CRFs, especially in the cases where the number of training data is small. One interesting result is that the M^{3}N and L_2-CRFs perform comparably. This is reasonable because as derived by Lebanon and Lafferty (2001) and noted by Globerson et al. (2007) that the L_2-regularized maximum likelihood estimation of CRFs has a similar convex dual as that of the M^{3}N, and the only difference is the loss they try to optimize, that is, CRFs optimize the log-loss while M^{3}N optimizes the hinge-loss. Another interesting observation is that when there are very few relevant features, L_1-M^{3}N performs the best (slightly better than LapMEDN); but as the number of relevant features increases LapMEDN performs slightly better than the L_1-M^{3}N. Finally, as the number of training data increases, all the algorithms consistently achieve better performance.

7.1.2 CORRELATED INPUT FEATURES

In reality, most data sets contain redundancies and the input features are usually correlated. So, we evaluate our models on synthetic data sets with correlated input features. We take the similar procedure as in generating the data sets with i.i.d. features to first generate 10 linear-chain CRF models. Then, each CRF is used to generate a data set that contain 1000 instances, each with 100 input features of which 30 are relevant to the output. The 30 relevant input features are partitioned into 10 groups. For the features in each group, we first draw a real-value from a standard normal distribution and then corrupt the feature with a random Gaussian noise to get 3 correlated features. The noise Gaussian has a zero mean and standard variance 0.05. Here and in all the remaining experiments, we use the sub-gradient method (Ratliff et al., 2007) to solve the QP problem in both M^{3}N and the variational learning algorithm of LapMEDN. We use the learning rate and complexity constant that are suggested by the authors, that is, $\alpha_t = \frac{1}{2\beta\sqrt{t}}$ and $C = 200\beta$, where β is a parameter we introduced to adjust α_t and C. We do K-fold CV on each data set and take the average over the

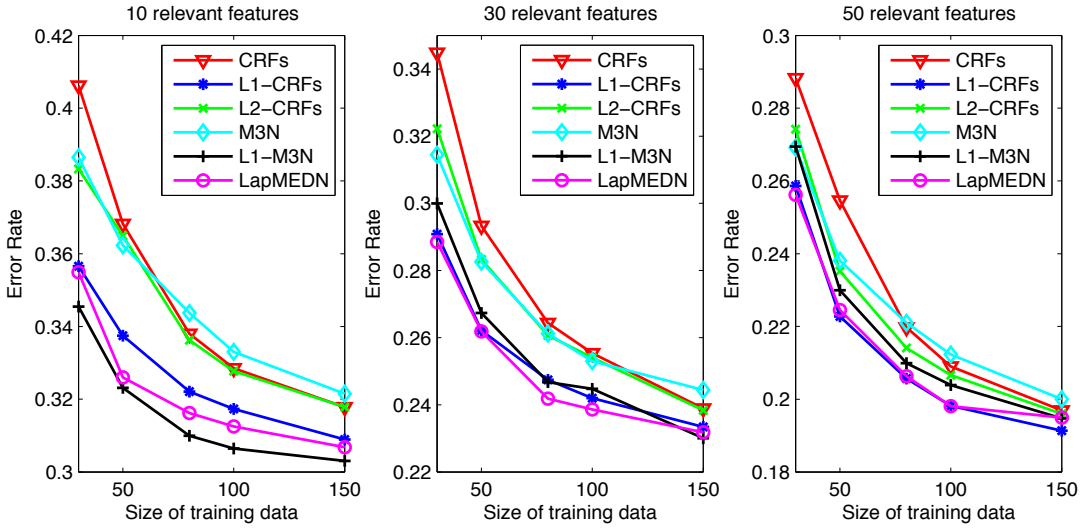


Figure 3: Evaluation results on data sets with i.i.d features.

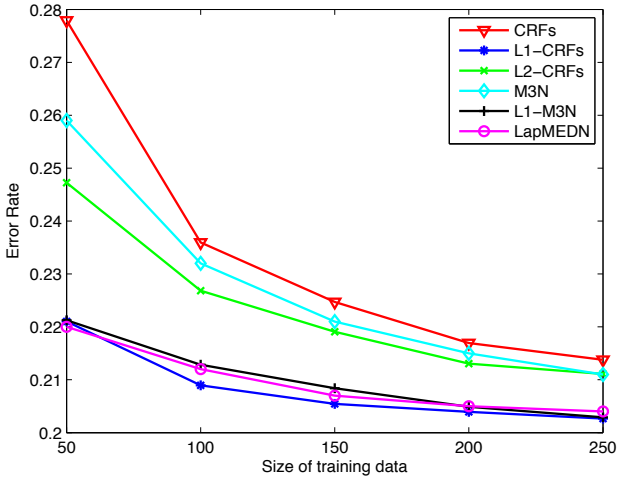


Figure 4: Results on data sets with 30 relevant features.

10 data sets as the final results. Like the method of Taskar et al. (2003), in each run we choose one part to do training and test on the rest K-1 parts. We vary K from 20, 10, 7, 5, to 4. In other words, we use 50, 100, about 150, 200, and 250 samples during the training. We use the same grid search to choose λ and β from $\{9, 16, 25, 36, 49, 64\}$ and $\{1, 10, 20, 30, 40, 50, 60\}$ respectively. Results are shown in Figure 4. We can get the same conclusions as in the previous results.

Figure 5 shows the true weights of the corresponding 200 state feature functions in the model that generates the first data set, and the average of estimated weights of these features under all competing models fitted from the first data set. All the averages are taken over 10 fold cross-validation. From the plots (2 to 7) of the average model weights, we can see that: for the last 140 state feature functions, which correspond to the last 70 irrelevant features, their average weights under LapMEDN (averaged posterior means $\mathbf{w}$ in this case), L_1-M^{3}N and L_1-CRFs are extremely small, while CRFs and L_2-CRFs can have larger values; for the first 60 state feature functions, which correspond to the 30 relevant features, the overall weight estimation under LapMEDN is similar to

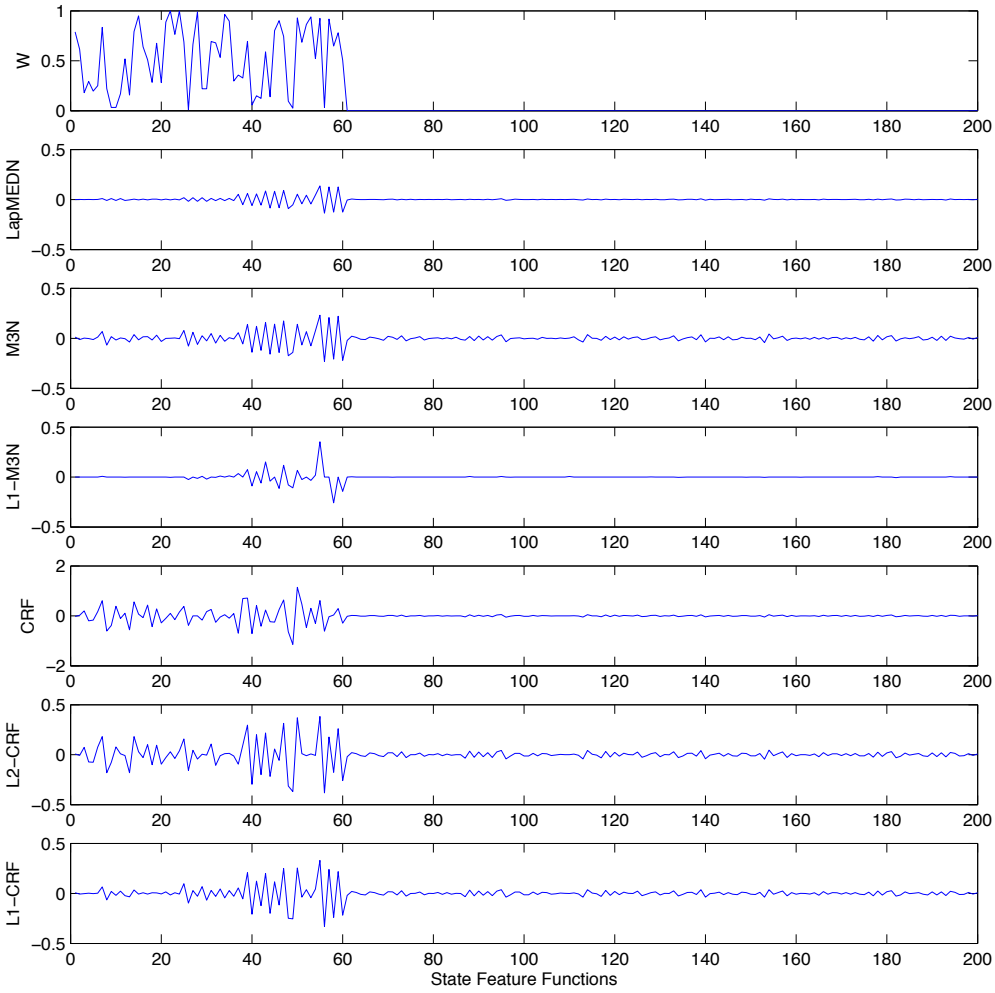


Figure 5: From top to bottom, plot 1 shows the weights of the state feature functions in the linear-chain CRF model from which the data are generated; plot 2 to plot 7 show the average weights of the learned LapMEDN, M^{3}N, L_1-M^{3}N, CRFs, L_2-CRFs, and L_1-CRFs over 10 fold CV, respectively.

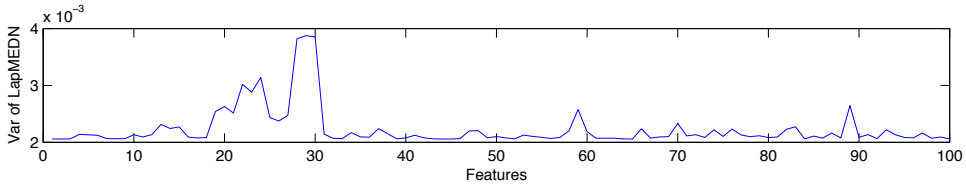


Figure 6: The average variances of the features on the first data set by LapMEDN.

that of the sparse L_1-CRFs and L_1-M^{3}N, but appear to exhibit more shrinkage. Noticeably, CRFs and L_2-CRFs both have more feature functions with large average weights. Note that all the models have quite different average weights from the model (see the first plot) that generates the data. This is because we use a stochastic procedure (i.e., Gibbs sampler) to assign labels to the generated data samples instead of using the labels that are predicted by the model that generates the data. In fact, if we use the model that generates the data to do prediction on its generated data, the error rate is about 0.5. Thus, the learned models, which get lower error rates, are different from the model that generates the data. Figure 6 shows the variances of the 100-dimensional input features (since the variances of the two feature functions that correspond to the same input feature are the same, we collapse each pair into one point) learned by LapMEDN. Again, the variances are the averages over 10 fold cross-validation. From the plot, we can see that the LapMEDN can recover the correlation among the features to some extend, e.g., for the first 30 correlated features, which are the relevant to the output, the features in the same group tend to have similar (average) variances in LapMEDN, whereas there is no such correlation among all the other features. From these observations in both Figure 5 and 6, we can conclude that LapMEDN can reasonably recover the sparse structures in the input data.

7.2 Real-World OCR Data Set

The OCR data set is partitioned into 10 subsets for 10-fold CV as in Taskar et al. (2003) and Ratliff et al. (2007). We randomly select N samples from each fold and put them together to do 10-fold CV. We vary N from 100, 150, 200, to 250, and denote the selected data sets by OCR100, OCR150, OCR200, and OCR250, respectively. On these data sets and the web data as in Section 7.4, our implementation of the cutting plane method for L_1-M^{3}N is extremely slow. The warm-start simplex method of MOSEK does not help either. For example, if we stop the algorithm with 600 iterations on OCR100, then it will take about 20 hours to finish the 10 fold CV. Even with more than 5 thousands of constraints in each training, the performance is still very bad (the error rate is about 0.45). Thus, we turn to an approximate projected sub-gradient method to solve the L_1-M^{3}N by combining the on-line subgradient method (Ratliff et al., 2007) and the efficient L_1-ball projection algorithm (Duchi et al., 2008). The projected sub-gradient method does not work so well as the cutting plane method on the synthetic data sets. That's why we use two different methods.

For $\beta = 4$ on OCR100 and OCR150, $\beta = 2$ on OCR200 and OCR250, and $\lambda = 36$, the results are shown in Figure 7. We can see that as the number of training instances increases, all the algorithms get lower error rates and smaller variances. Generally, the LapMEDN consistently outperforms all the other models. M^{3}N outperforms the standard, non-regularized, CRFs and the L_1-CRFs. Again, L_2-CRFs perform comparably with M^{3}N. This is a bit surprising but still reasonable due to the understanding of their only difference on the loss functions (Globerson et al., 2007) as we have stated. By examining the prediction accuracy during the learning, we can see an obvious over-fitting in CRFs and L_1-CRFs as shown in Figure 8. In contrast, L_2-CRFs are very robust. This is because unlike the synthetic data sets, features in real-world data are usually not completely irrelevant. In this case, putting small weights to zero as in L_1-CRFs will hurt generalization ability and also lead to instability to regularization constants as shown later. Instead, L_2-CRFs do not put small weights to zero but shrink them towards zero as in the LapMEDN. The non-regularized maximum likelihood estimation can easily lead to over-fitting too. For the two sparse models, the results suggest the

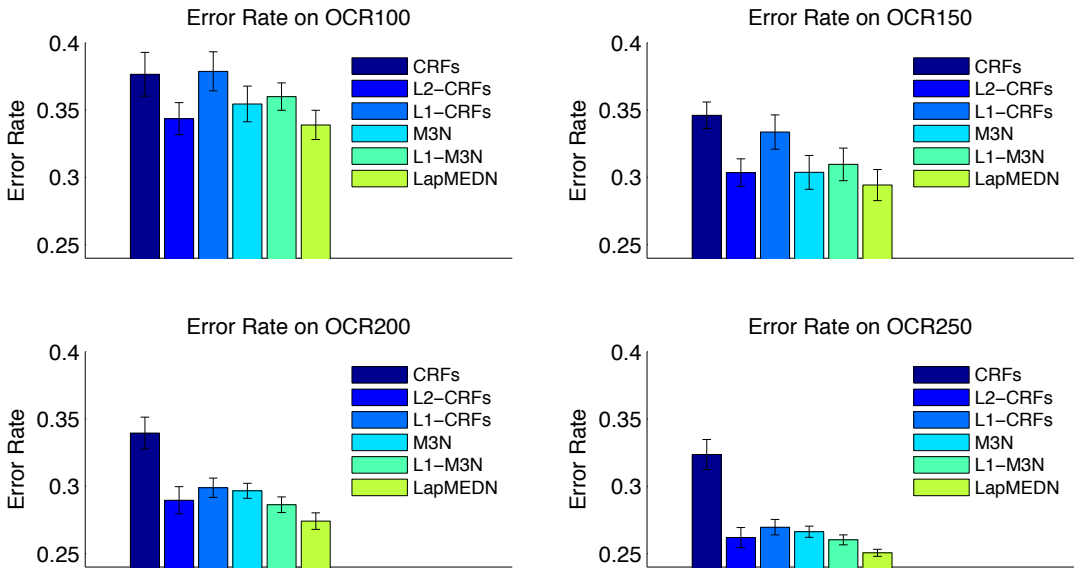


Figure 7: Evaluation results on OCR data set with different numbers of selected data.

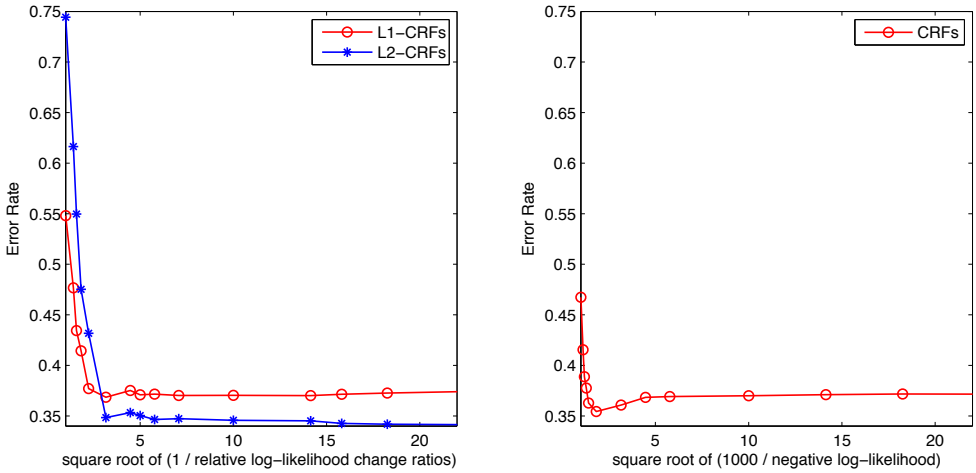


Figure 8: The error rates of CRF models on test data during the learning. For the left plot, the horizontal axis is $\sqrt{1/ratioLL}$, where *ratioLL* is the relative change ratios of the log-likelihood and from left to right, the change ratios are 1, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05, 0.04, 0.03, 0.02, 0.01, 0.005, 0.004, 0.003, 0.002, 0.001, 0.0005, 0.0004, 0.0003, 0.0002, 0.0001, and 0.00005; for the right plot, the horizontal axis is $\sqrt{1000/negLL}$, where *negLL* is the negative log-likelihood, and from left to right *negLL* are 1000, 800, 700, 600, 500, 300, 100, 50, 30, 10, 5, 3, 1, 0.5, 0.3, 0.1, 0.05, 0.03, 0.01, 0.005, 0.003, and 0.002.

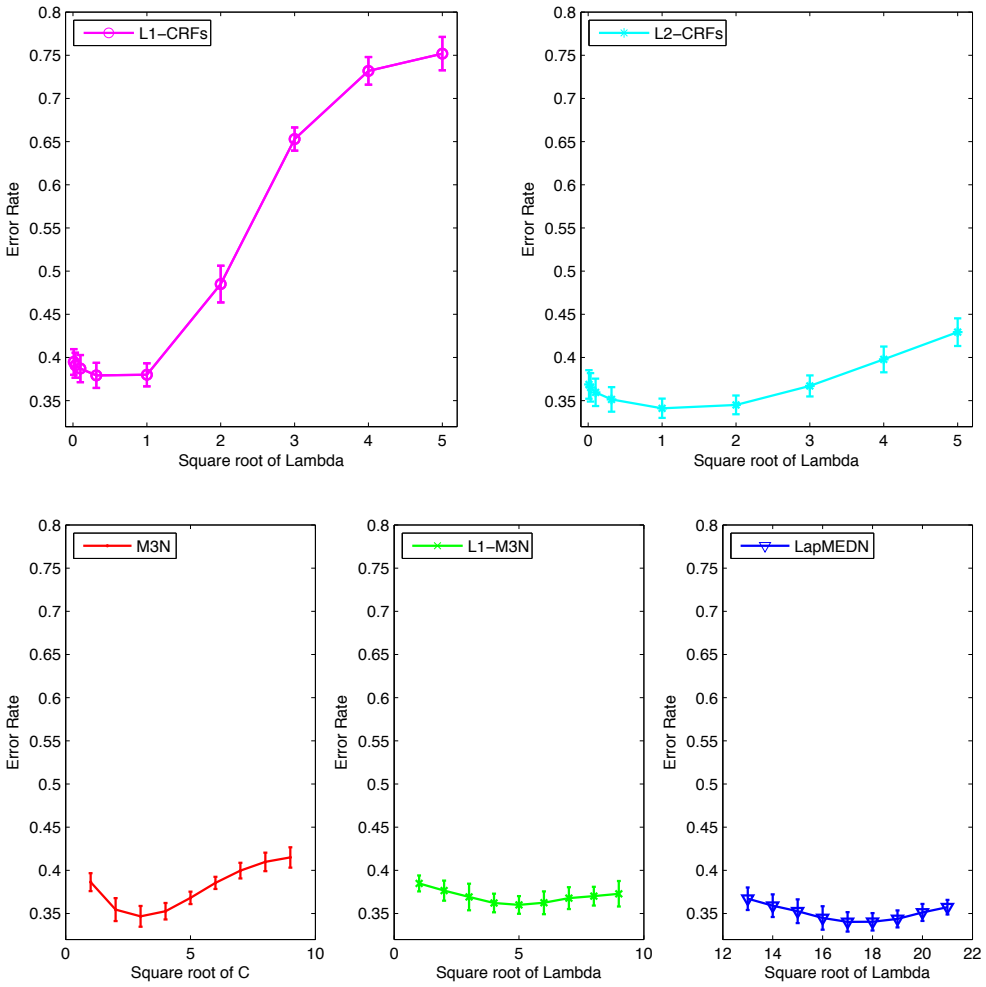


Figure 9: Error rates of different models on OCR100 with different regularization constants. The regularization constant is the parameter C for M^{3}N, and for all the other models, it is the parameter λ. From left to right, the regularization constants for the two regularized CRFs (above plot) are $0.0001, 0.001, 0.01, 0.1, 1, 4, 9, 16$, and 25; for M^3N and LapMEDN, the regularization constants are $k^2,\ 1 \leq k \leq 9$; and for L_1-M^{3}N, the constants are $k^2,\ 13 \leq k \leq 21$.

potential advantages of L_1-norm regularized M^{3}N, which are consistently better than the L_1-CRFs. Furthermore, as we shall see later, L_1-M^{3}N is more stable than the L_1-CRFs.

7.3 Sensitivity to Regularization Constants

Figure 9 shows the error rates of the models in question on the data set OCR100 over different magnitudes of the regularization constants. For M^{3}N, the regularization constant is the parameter

C, and for all the other models, the regularization constant is the parameter λ. When the λ changes, the parameter C in LapMEDN and L_1-M^{3}N is fixed at the unit 1.

From the results, we can see that the L_1-CRFs are quite sensitive to the regularization constants. However, L_2-CRFs, M^{3}N, L_1-M^{3}N and LapMEDN are much less sensitive. LapMEDN and L_1-M^{3}N are the most stable models. The stability of LapMEDN is due to the posterior weighting instead of hard-thresholding to set small weights to zero as in the L_1-CRFs. One interesting observation is that the max-margin based L_1-M^{3}N is much more stable compared to the L_1-norm regularized CRFs. One possible reason is that like LapMEDN, L_1-M^{3}N enjoys both the primal and dual sparsity, which makes it less sensitive to outliers; whereas the L_1-CRF is only primal sparse.

7.4 Real-World Web Data Extraction

The last experiments are conducted on the real world web data extraction as extensively studied by Zhu et al. (2008a). Web data extraction is a task to identify interested information from web pages. Each sample is a data record or an entire web page which is represented as a set of HTML elements. One striking characteristic of web data extraction is that various types of structural dependencies between HTML elements exist, e.g. the HTML tag tree or the Document Object Model (DOM) structure is itself hierarchical. In the work of Zhu et al. (2008a), hierarchical CRFs are shown to have great promise and achieve better performance than flat models like linear-chain CRFs (Lafferty et al., 2001). One method to construct a hierarchical model is to first use a parser to construct a so called vision tree. Then, based on the vision tree, a hierarchical model can be constructed accordingly to extract the interested attributes, e.g. a product's name, image, price, description, etc. See the paper (Zhu et al., 2008a) for an example of the vision tree and the corresponding hierarchical model. In such a hierarchical extraction model, inner nodes are useful to incorporate long distance dependencies, and the variables at one level are refinements of the variables at upper levels.

In these experiments,[6] we identify product items for sale on the Web. For each product item, four attributes—*Name*, *Image*, *Price*, and *Description* are extracted. We use the data set that is built with web pages generated by 37 different templates (Zhu et al., 2008a). For each template, there are 5 pages for training and 10 for testing. We evaluate all the methods on the *record level*, that is, we assume that data records are given, and we compare different models on the accuracy of extracting attributes in the given records. In the 185 training pages, there are 1585 data records in total; in the 370 testing pages, 3391 data records are collected. As for the evaluation criteria, we use the two comprehensive measures, that is, average F1 and block instance accuracy. As defined by Zhu et al. (2008a), average F1 is the average value of the F1 scores of the four attributes, and block instance accuracy is the percent of data records whose *Name*, *Image*, and *Price* are all correctly identified.

We randomly select $m = 5, 10, 15, 20, 30, 40, \text{or}, 50$ percent of the training records as training data, and test on all the testing records. For each m, 10 independent experiments were conducted and the average performance is summarized in Figure 10. From the results, we can see that all: first, the models (especially the max-margin models, that is, M^{3}N, L_1-M^{3}N, and LapMEDN) with regularization (i.e., L_1-norm, L_2-norm, or the entropic regularization of LapMEDN) can significantly outperform the un-regularized CRFs. Second, the max-margin models generally outperform the conditional likelihood-based models (i.e., CRFs, L_2-CRFs, and L_1-CRFs). Third, the LapMEDN perform comparably with the L_1-M^{3}N, which enjoys both dual and primal sparsity as the LapMEDN, and

6. These experiments are slightly different from those by Zhu et al. (2008a). Here, we introduce more general feature functions based on the content and visual features as used by Zhu et al. (2008a).

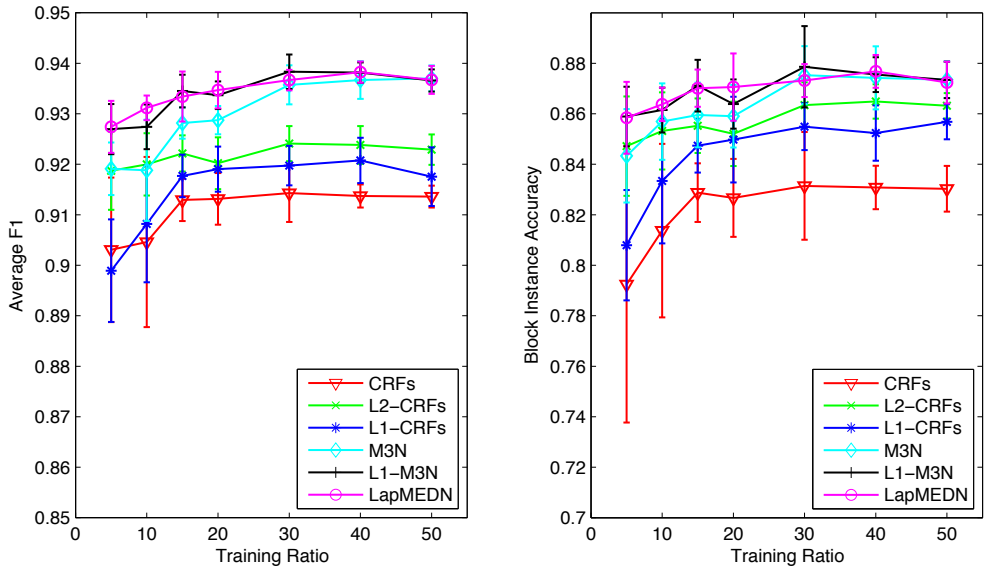


Figure 10: The average F1 values and block instance accuracy on web data extraction with different number of training data.

outperforms all other models, especially when the number of training data is small. Finally, as in the previous experiments on OCR data, the L_1-M^{3}N generally outperforms the L_1-CRFs, which suggests the potential promise of the max-margin based L_1-M^{3}N.

8. Related Work

Our work is motivated by the maximum entropy discrimination (MED) method proposed by Jaakkola et al. (1999), which integrates SVM and entropic regularization to obtain an averaging maximum margin model for classification. The MaxEnDNet model presented is essentially a structured version of MED built on M^{3}N—the so called "structured SVM". As we presented in this paper, this extension leads to a substantially more flexible and powerful new paradigm for structured discriminative learning and prediction, which enjoys a number of advantages such as model averaging, primal and dual sparsity, accommodation of latent generative structures, but at the same time raises new algorithmic challenges in inference and learning.

Related to our approach, a sparse Bayesian learning framework has been proposed to find sparse and robust solutions to regression and classification. One example along this line is the relevance vector machine (RVM) (Tipping, 2001). The RVM was proposed based on SVM. But unlike SVM which directly optimizes on the margins, RVM defines a likelihood function from the margins with a Gaussian distribution for regression and a logistic sigmoid link function for classification and then does *type-II maximum likelihood* estimation, that is, RVM maximizes the *marginal likelihood*. Although called *sparse Bayesian learning* (Figueiredo, 2001; Eyheramendy et al., 2003), as shown by Kaban (2007) the sparsity is actually due to the MAP estimation. The similar ambiguity of RVM is justified by Wipf et al. (2003). Unlike these approaches, we adhere to a full Bayesian-style principle and learn a distribution of predictive models by optimizing a generalized maximum entropy under a set of the *expected* margin constraints. By defining likelihood functions with margins, similar

Bayesian interpretations of both binary and multi-class SVM were studied by Sollich (2002) and Zhang and Jordan (2006).

The hierarchical interpretation of the Laplace prior has been explored in a number of contexts in the literature. Based on this interpretation, a Jeffrey's non-informative second-level hyper-prior was proposed by Figueiredo (2001), with an EM algorithm developed to find the MAP estimate. The advantage of the Jeffrey's prior is that it is parameter-free. But as shown by Eyheramendy et al. (2003) and Kaban (2007), usually no advantage is achieved by using the Jeffrey's hyper-prior over the Laplace prior. A gamma hyper-prior was used by Tipping (2001) in place of the second-level exponential as in the hierarchical interpretation of the Laplace prior.

To encourage sparsity in SVM, two strategies have been used. The first one is to replace the L_2-norm by an L_1-norm of the weights (Bennett and Mangasarian, 1992; Zhu et al., 2004). The second strategy is to explicitly add a cardinality constraint on the weights. This will lead to a hard non-convex optimization problem; thus relaxations must be applied (Chan et al., 2007). Under the maximum entropy discrimination models, feature selection was studied by Jebara and Jaakkola (2000) by introducing a set of structural variables. Recently, a smooth posterior shrinkage effect was shown by Jebara (2009), which is similar to our entropic regularization effect. However, an analysis of their connections and differences is still not obvious.

Although the parameter distribution $p(\mathbf{w})$ in Theorem 2 has a similar form as that of the Bayesian Conditional Random Fields (BCRFs) (Qi et al., 2005), MaxEnDNet is fundamentally different from BCRFs as we have stated. Dredze et al. (2008) present an interesting confidence-weighted linear classification method, which automatically estimates the mean and variance of model parameters in online learning. The procedure is similar to (but indeed different from) our variational Bayesian method of Laplace MaxEnDNet.

Finally, some of the results shown in this paper can be also found in our recent conference papers (Zhu et al., 2008b; Zhu and Xing, 2009).

9. Conclusions and Future Work

To summarize, we have presented a general theory of maximum entropy discrimination Markov networks for structured input/output learning and prediction. This formalism offers a formal paradigm for integrating both generative and discriminative principles and the Bayesian regularization techniques for learning structured prediction models. It subsumes popular methods such as support vector machines, maximum entropy discrimination models (Jaakkola et al., 1999), and maximum margin Markov networks as special cases, and therefore inherits all the merits of these techniques.

The MaxEnDNet model offers a number of important advantages over conventional structured prediction methods, including: 1) model averaging, which leads to a PAC-Bayesian bound on generalization error; 2) entropic regularization over max-margin learning, which can be leveraged to learn structured prediction models that are simultaneously primal and dual sparse; and 3) latent structures underlying the structured input/output variables, which enables better incorporation of domain knowledge in model design and semi-supervised learning based on partially labeled data. In this paper, we have discussed in detail the first two aspects, and the third aspect was explored in (Zhu et al., 2008c). We have also shown that certain instantiations of the MaxEnDNet model, such as the LapMEDN that achieves primal and dual sparsity, can be efficiently trained based on an iterative optimization scheme that employs existing techniques such as the variational Bayes approximation and the convex optimization procedures that solve the standard M^3N. We demonstrated that on syn-

thetic data the LapMEDN can recover the sparse model as well as the sparse L_1-regularized MAP estimation, and on real data sets LapMEDN can achieve superior performance.

Overall, we believe that the MaxEnDNet model can be extremely general and adaptive, and it offers a promising new framework for building more flexible, generalizable, and large scale structured prediction models that enjoy the benefits from both generative and discriminative modeling principles. While exploring novel instantiations of this model will be an interesting direction to pursue, development of more efficient learning algorithms, formulation of tighter but easy to solve convex relaxations, and adapting this model to challenging applications such as statistical machine translation, and structured associations of genome markers to complex disease traits could also lead to fruitful results. Finally, the basic principle of MaxEnDNet can be generalized to directed graphical models. The MedLDA model (Zhu et al., 2009a) for discriminative topic modeling represents our first successful attempt along this direction.

Acknowledgments

Jun Zhu was with the Department of Computer Science and Technology, Tsinghua University. This work was done while Jun Zhu was visiting the SAILING Lab directed by Eric Xing at Carnegie Mellon University, under a visiting scholarship sponsored by the National Natural Science Foundation of China. The authors would like to thank Andrew Bagnell for sharing their implementation of the sub-gradient algorithm, Ivor Tsang and André Martins for inspiring discussions. Eric Xing is supported by NSF grants CCF-0523757, DBI-0546594, IIS-0713379, DBI- 0640543, and a Sloan Research Fellowship in Computer Science. Jun Zhu was supported by the National Natural Science Foundation of China, Grant. No. 60321002; the National Key Foundation R&D Projects, Grant No. 2004CB318108 and 2007CB311003; and Microsoft Fellowship.

Appendix A. L_1-M^{3}N and its Lagrange-Dual

Based on the L_1-norm regularized SVM (Zhu et al., 2004; Bennett and Mangasarian, 1992), a straightforward formulation of L_1-M^{3}N is as follows,

$$\begin{aligned}&\min_{\mathbf{w},\xi} \ \frac{1}{2}\|\mathbf{w}\| + C\sum_{i=1}^{N}\xi_i \\ &\text{s.t.} \ \ \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i;\ \xi_i \geq 0,\ \forall i,\ \forall \mathbf{y} \neq \mathbf{y}^i\end{aligned}$$

where $\|.\|$ is the L_1-norm. $\Delta\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^i,\mathbf{y}^i) - \mathbf{f}(\mathbf{x}^i,\mathbf{y})$, and $\Delta\ell_i(\mathbf{y})$ is a loss function. Another equivalent formulation[7] is as follows:

$$\begin{aligned}&\min_{\mathbf{w},\xi} \ C\sum_{i=1}^{N}\xi_i \\ &\text{s.t.} \ \ \begin{cases} \|\mathbf{w}\| \leq \lambda \\ \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i;\ \xi_i \geq 0,\ \forall i,\ \forall \mathbf{y} \neq \mathbf{y}^i. \end{cases}\end{aligned}$$

7. See Taskar et al. (2006) for the transformation technique.

To derive the convex dual problem, we introduce a dual variable $\alpha_i(\mathbf{y})$ for each constraint in the former formulation and form the Lagrangian as follows,

$$L(\alpha, \mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\| + C\sum_{i=1}^{N}\xi_i - \sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})(\mathbf{w}^\top\Delta\mathbf{f}_i(\mathbf{y}) - \Delta\ell_i(\mathbf{y}) + \xi_i).$$

By definition, the Lagrangian dual is,

$$\begin{aligned} L^\star(\alpha) &= \inf_{\mathbf{w},\xi} L(\alpha, \mathbf{w}, \xi) \\ &= \inf_{\mathbf{w}}\Big[\frac{1}{2}\|\mathbf{w}\| - \sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\mathbf{w}^\top\Delta\mathbf{f}_i(\mathbf{y})\Big] + \inf_{\xi}\Big[C\sum_{i=1}^{N}\xi_i - \sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\xi_i\Big] + \ell \\ &= -\sup_{\mathbf{w}}\Big[\mathbf{w}^\top(\sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})) - \frac{1}{2}\|\mathbf{w}\|\Big] - \sup_{\xi}\Big[\sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\xi_i - C\sum_{i=1}^{N}\xi_i\Big] + \ell, \end{aligned}$$

where $\ell = \sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y})$.

Again, by definition, the first term on the right-hand side is the convex conjugate of $\phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|$ and the second term is the conjugate of $U(\xi) = C\sum_{i=1}^{N}\xi_i$. It is easy to show that,

$$\phi^\star(\alpha) = \mathbb{I}_\infty(|\sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\Delta\mathbf{f}_i^k(\mathbf{y})| \leq \frac{1}{2},\ \forall 1 \leq k \leq K),$$

and

$$U^\star(\alpha) = \mathbb{I}_\infty(\sum_{\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y}) \leq C,\ \forall i),$$

where as defined before $\mathbb{I}_\infty(\cdot)$ is an indicator function that equals zero when its argument is true and infinity otherwise. $\Delta\mathbf{f}_i^k(\mathbf{y}) = f_k(\mathbf{x}^i, \mathbf{y}^i) - f_k(\mathbf{x}^i, \mathbf{y})$.

Therefore, we get the dual problem as follows,

$$\begin{aligned} \max_{\alpha}\ & \sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y}) \\ \text{s.t. }\ & |\sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\Delta\mathbf{f}_i^k(\mathbf{y})| \leq \frac{1}{2},\ \forall k \\ & \sum_{\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y}) \leq C,\ \forall i. \end{aligned}$$

Appendix B. Proofs of Theorems and Corollaries

In this section, we prove the theorems and corollaries.

B.1 Proof of Theorem 2

Proof As we have stated, P1 is a convex program and satisfies the Slater's condition. To compute its convex dual, we introduce a non-negative dual variable $\alpha_i(\mathbf{y})$ for each constraint in $\mathcal{F}_1$ and another

non-negative dual variable c for the normalization constraint $\int p(\mathbf{w})\,\mathrm{d}\mathbf{w} = 1$. This gives rise to the following Lagrangian:

$$\begin{aligned} L(p(\mathbf{w}),\xi,\alpha,c) \quad &= KL(p(\mathbf{w})||p_0(\mathbf{w})) + U(\xi) + c(\int p(\mathbf{w})\,\mathrm{d}\mathbf{w} - 1) \\ &- \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})(\int p(\mathbf{w})[\Delta F_i(\mathbf{y};\mathbf{w}) - \Delta\ell_i(\mathbf{y})]\,\mathrm{d}\mathbf{w} + \xi_i). \end{aligned}$$

The Lagrangian dual function is defined as $L^\star(\alpha,c) \triangleq \inf_{p(\mathbf{w});\xi} L(p(\mathbf{w}),\xi,\alpha,c)$. Taking the derivative of L w.r.t $p(\mathbf{w})$, we get,

$$\frac{\partial L}{\partial p(\mathbf{w})} = 1 + c + \log\frac{p(\mathbf{w})}{p_0(\mathbf{w})} - \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})[\Delta F_i(\mathbf{y};\mathbf{w}) - \Delta\ell_i(\mathbf{y})].$$

Setting the derivative to zero, we get the following expression of distribution $p(\mathbf{w})$,

$$p(\mathbf{w}) = \frac{1}{Z(\alpha)} p_0(\mathbf{w}) \exp\{\sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})[\Delta F_i(\mathbf{y};\mathbf{w}) - \Delta\ell_i(\mathbf{y})]\},$$

where $Z(\alpha) \triangleq \int p_0(\mathbf{w}) \exp\{\sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})[\Delta F_i(\mathbf{y};\mathbf{w}) - \Delta\ell_i(\mathbf{y})]\}\,\mathrm{d}\mathbf{w}$ is a normalization constant and $c = -1 + \log Z(\alpha)$.

Substituting $p(\mathbf{w})$ into $L^\star$, we obtain,

$$\begin{aligned} L^\star(\alpha,c) &= \inf_{p(\mathbf{w});\xi} (-\log Z(\alpha) + U(\xi) - \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\xi_i) \\ &= -\log Z(\alpha) + \inf_{\xi} (U(\xi) - \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\xi_i) \\ &= -\log Z(\alpha) - \sup_{\xi} (\sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\xi_i - U(\xi)) \\ &= -\log Z(\alpha) - U^\star(\alpha), \end{aligned}$$

which is the objective in the dual problem D1. The $\{\alpha_i(\mathbf{y})\}$ derived from D1 lead to the optimum $p(\mathbf{w})$ according to Equation (3). ■

B.2 Proof of Theorem 3

Proof Replacing $p_0(\mathbf{w})$ and $\Delta F_i(\mathbf{y};\mathbf{w})$ in Equation (3) with $\mathcal{N}(\mathbf{w}|0,I)$ and $\mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y})$ respectively, we can obtain the following closed-form expression of the $Z(\alpha)$ in $p(\mathbf{w})$:

$$\begin{aligned} Z(\alpha) \quad &\triangleq \int \mathcal{N}(\mathbf{w}|0,I) \exp\Big\{\sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})[\mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) - \Delta\ell_i(\mathbf{y})]\Big\}\,\mathrm{d}\mathbf{w} \\ &= \int (2\pi)^{-\frac{K}{2}} \exp\Big\{-\frac{1}{2}\mathbf{w}^\top\mathbf{w} + \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})[\mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) - \Delta\ell_i(\mathbf{y})]\Big\}\,\mathrm{d}\mathbf{w} \\ &= \exp\Big(-\sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y}) + \frac{1}{2}\|\sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})\|^2\Big). \end{aligned}$$

Substituting the normalization factor into the general dual problem D1, we get the dual problem of Gaussian MaxEnDNet. As we have stated, the constraints $\sum_{\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y}) = C$ are due to the conjugate of $U(\xi) = C\sum_i \xi_i$.

For prediction, again replacing $p_0(\mathbf{w})$ and $\Delta F_i(\mathbf{y};\mathbf{w})$ in Equation (3) with $\mathcal{N}(\mathbf{w}|0,I)$ and $\mathbf{w}^\top\Delta\mathbf{f}_i(\mathbf{y})$ respectively, we can get $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mu,I)$, where $\mu = \sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})$. Substituting $p(\mathbf{w})$ into the predictive function h_1, we can get $h_1(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})}\mu^\top\mathbf{f}(\mathbf{x},\mathbf{y}) = (\sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y}))^\top\mathbf{f}(\mathbf{x},\mathbf{y})$, which is identical to the prediction rule of the standard M^{3}N (Taskar et al., 2003) because the dual parameters are achieved by solving the same dual problem. ■

B.3 Proof of Corollary 4

Proof Suppose $(p^\star(\mathbf{w}),\xi^\star)$ is the optimal solution of P1, then we have: for any $(p(\mathbf{w}),\xi),\ p(\mathbf{w}) \in \mathcal{F}_1$ and $\xi \geq 0$,

$$KL(p^\star(\mathbf{w})||p_0(\mathbf{w})) + U(\xi^\star) \leq KL(p(\mathbf{w})||p_0(\mathbf{w})) + U(\xi).$$

From Theorem 3, we conclude that the optimum predictive parameter distribution is $p^\star(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mu^\star,I)$. Since $p_0(\mathbf{w})$ is also normal, for any distribution $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mu,I)$[8] with several steps of algebra it is easy to show that $KL(p(\mathbf{w})|p_0(\mathbf{w})) = \frac{1}{2}\mu^\top\mu$. Thus, we can get: for any $(\mu,\xi),\ \mu \in \{\mu:\ \mu^\top\Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i,\ \forall i,\ \forall\mathbf{y}\neq\mathbf{y}^i\}$ and $\xi \geq 0$,

$$\frac{1}{2}(\mu^\star)^\top(\mu^\star) + U(\xi^\star) \leq \frac{1}{2}\mu^\top\mu + U(\xi^\star),$$

which means the mean of the optimum posterior distribution under a Gaussian MaxEnDNet is achieved by solving a primal problem as stated in the Corollary. ■

B.4 Proof of Corollary 6

Proof The proof follows the same structure as the above proof of Corollary 4. Here, we only present the derivation of the KL-divergence under the Laplace MaxEnDNet.

Theorem 2 shows that the general posterior distribution is $p(\mathbf{w}) = \frac{1}{Z(\alpha)}p_0(\mathbf{w})\exp(\mathbf{w}^\top\eta - \sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y}))$ and $Z(\alpha) = \exp(-\sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y}))\prod_{k=1}^K\frac{\lambda}{\lambda-\eta_k^2}$ for the Laplace MaxEnDNet as shown in Equation (5). Use the definition of KL-divergence and we can get:

$$KL(p(\mathbf{w})|p_0(\mathbf{w})) = \langle\mathbf{w}\rangle_p^\top\eta - \sum_{k=1}^K\log\frac{\lambda}{\lambda-\eta_k^2} = \sum_{k=1}^K\mu_k\eta_k - \sum_{k=1}^K\log\frac{\lambda}{\lambda-\eta_k^2},$$

Corollary 7 shows that $\mu_k = \frac{2\eta_k}{\lambda-\eta_k^2},\ \forall 1\leq k\leq K$. Thus, we get $\frac{\lambda}{\lambda-\eta_k^2} = \frac{\lambda\mu_k}{2\eta_k}$ and a set of equations: $\mu_k\eta_k^2 + 2\eta_k - \lambda\mu_k = 0,\ \forall 1\leq k\leq K$. To solve these equations, we consider two cases. First, if $\mu_k = 0$, then $\eta_k = 0$. Second, if $\mu_k \neq 0$, then we can solve the quadratic equation to get

8. Although $\mathcal{F}_1$ is much richer than the set of normal distributions with an identity covariance matrix, Theorem 3 shows that the solution is a restricted normal distribution. Thus, it suffices to consider only these normal distributions in order to learn the mean of the optimum distribution. The similar argument applies to the proof of Corollary 6.

η_k: $\eta_k = \frac{-1\pm\sqrt{1+\lambda\mu_k^2}}{\mu_k}$. The second solution includes the first one since we can show that when $\mu_k \to 0$, $\frac{-1\pm\sqrt{1+\lambda\mu_k^2}}{\mu_k} \to 0$ by using the *L'Hospital's Rule*. Thus, we get:

$$\mu_k\eta_k = -1 \pm \sqrt{\lambda\mu_k^2+1}.$$

Since $\eta_k^2 < \lambda$ (otherwise the problem is not bounded), $\mu_k\eta_k$ is always positive. Thus, only the solution $\mu_k\eta_k = -1+\sqrt{1+\lambda\mu_k^2}$ is feasible. So, we get:

$$\frac{\lambda}{\lambda-\eta_k^2} = \frac{\lambda\mu_k^2}{2(\sqrt{\lambda\mu_k^2+1}-1)} = \frac{\sqrt{\lambda\mu_k^2+1}+1}{2},$$

and

$$\begin{aligned} KL(p(\mathbf{w})|p_0(\mathbf{w})) &= \sum_{k=1}^{K}\Big(\sqrt{\lambda\mu_k^2+1} - \log\frac{\sqrt{\lambda\mu_k^2+1}+1}{2}\Big) - K \\ &= \sqrt{\lambda}\sum_{k=1}^{K}\Big(\sqrt{\mu_k^2+\frac{1}{\lambda}} - \frac{1}{\sqrt{\lambda}}\log\frac{\sqrt{\lambda\mu_k^2+1}+1}{2}\Big) - K. \end{aligned}$$

Applying the same arguments as in the above proof of Corollary 4 and using the above result of the KL-divergence, we get the problem in Corollary 6, where the constant $-K$ is ignored. The margin constraints defined with the mean μ are due to the linearity assumption of the discriminant functions. ■

B.5 Proof of Theorem 8

We follow the same structure as the proof of PAC-Bayes bound for binary classifier (Langford et al., 2001) and employ the similar technique to generalize to multi-class problems (Schapire et al., 1998). Recall that the output space is $\mathcal{Y}$, and the base discriminant function is $F(\,\cdot\,;\mathbf{w}) \in \mathcal{H} : \mathcal{X}\times\mathcal{Y} \to [-c,c]$, where $c>0$ is a constant. Our averaging model is specified by $h(\mathbf{x},\mathbf{y}) = \langle F(\mathbf{x},\mathbf{y};\mathbf{w})\rangle_{p(\mathbf{w})}$. We define the margin of an example $(\mathbf{x},\mathbf{y})$ for such a function h as,

$$M(h,\mathbf{x},\mathbf{y}) = h(\mathbf{x},\mathbf{y}) - \max_{\mathbf{y}'\neq\mathbf{y}} h(\mathbf{x},\mathbf{y}'). \tag{12}$$

Thus, the model h makes a wrong prediction on $(\mathbf{x},\mathbf{y})$ only if $M(h,\mathbf{x},\mathbf{y}) \le 0$. Let Q be a distribution over $\mathcal{X}\times\mathcal{Y}$, and let $\mathcal{D}$ be a sample of N examples independently and randomly drawn from Q. With these definitions, we have the PAC-Bayes theorem. For easy reading, we copy the theorem in the following:

Theorem 8 (PAC-Bayes Bound of MaxEnDNet) *Let p_0 be any continuous probability distribution over $\mathcal{H}$ and let $\delta\in(0,1)$. If $F(\,\cdot\,;\mathbf{w}) \in \mathcal{H}$ is bounded by $\pm c$ as above, then with probability*

at least $1-\delta$*, for a random sample* $\mathcal{D}$ *of* N *instances from* Q*, for every distribution* p *over* $\mathcal{H}$*, and for all margin thresholds* $\gamma > 0$*:*

$$\Pr_Q(M(h,\mathbf{x},\mathbf{y}) \leq 0) \leq \Pr_{\mathcal{D}}(M(h,\mathbf{x},\mathbf{y}) \leq \gamma) + O\Big(\sqrt{\frac{\gamma^{-2}KL(p||p_0)\ln(N|\mathcal{Y}|) + \ln N + \ln \delta^{-1}}{N}}\Big),$$

where $\Pr_Q(.)$ *and* $\Pr_{\mathcal{D}}(.)$ *represent the probabilities of events over the true distribution* Q*, and over the empirical distribution of* $\mathcal{D}$*, respectively.*

Proof Let m be any natural number. For every distribution p, we independently draw m base models (i.e., discriminant functions) $F_i \sim p$ at random. We also independently draw m variables $\mu_i \sim U([-c,c])$, where U denote the uniform distribution. We define the binary functions $g_i : \mathcal{X} \times \mathcal{Y} \to \{-c,+c\}$ by:

$$g_i(\mathbf{x},\mathbf{y};F_i,\mu_i) = 2cI(\mu_i < F_i(\mathbf{x},\mathbf{y})) - c.$$

With the F_i, μ_i, and g_i, we define $\mathcal{H}_m$ as,

$$\mathcal{H}_m = \{f : (\mathbf{x},\mathbf{y}) \mapsto \frac{1}{m}\sum_{i=1}^{m} g_i(\mathbf{x},\mathbf{y};F_i,\mu_i) | F_i \in \mathcal{H}, \mu_i \in [-c,c]\}.$$

We denote the distribution of f over the set $\mathcal{H}_m$ by p^m. For a fixed pair $(\mathbf{x},\mathbf{y})$, the quantities $g_i(\mathbf{x},\mathbf{y};F_i,\mu_i)$ are i.i.d bounded random variables with the mean:

$$\begin{aligned}
\langle g_i(\mathbf{x},\mathbf{y};F_i,\mu_i)\rangle_{F_i\sim p,\mu_i\sim U[-c,c]} &= \langle (+c)p[\mu_i \leq F_i(\mathbf{x},\mathbf{y})|F_i] + (-c)p[\mu_i > F_i(\mathbf{x},\mathbf{y})|F_i]\rangle_{F_i\sim p} \\
&= \langle \frac{1}{2c}c(c+F_i(\mathbf{x},\mathbf{y})) - \frac{1}{2c}c(c-F_i(\mathbf{x},\mathbf{y}))\rangle_{F_i\sim p} \\
&= h(\mathbf{x},\mathbf{y}).
\end{aligned}$$

Therefore, $\langle f(\mathbf{x},\mathbf{y})\rangle_{f\sim p^m} = h(\mathbf{x},\mathbf{y})$. Since $f(\mathbf{x},\mathbf{y})$ is the average over m i.i.d bounded variables, Hoeffding's inequality applies. Thus, for every $(\mathbf{x},\mathbf{y})$,

$$\Pr_{f\sim p^m}[f(\mathbf{x},\mathbf{y}) - h(\mathbf{x},\mathbf{y}) > \xi] \leq e^{-\frac{m}{2c^2}\xi^2}.$$

For any two events A and B, we have the inequality,

$$\Pr(A) = \Pr(A,B) + \Pr(A,\bar{B}) \leq \Pr(B) + \Pr(\bar{B}|A).$$

Thus, for any $\gamma > 0$ we have

$$\Pr_Q[M(h,\mathbf{x},\mathbf{y}) \leq 0] \leq \Pr_Q[M(f,\mathbf{x},\mathbf{y}) \leq \frac{\gamma}{2}] + \Pr_Q[M(f,\mathbf{x},\mathbf{y}) > \frac{\gamma}{2} | M(h,\mathbf{x},\mathbf{y}) \leq 0], \tag{13}$$

where the left hand side does not depend on f. We take the expectation w.r.t $f \sim p^m$ on both sides and can get

$$\Pr_Q[M(h,\mathbf{x},\mathbf{y}) \leq 0] \leq \quad \langle \Pr_Q[M(f,\mathbf{x},\mathbf{y}) \leq \frac{\gamma}{2}] \rangle_{f\sim p^m} + \langle \Pr_Q[M(f,\mathbf{x},\mathbf{y}) > \frac{\gamma}{2} | M(h,\mathbf{x},\mathbf{y}) \leq 0] \rangle_{f\sim p^m}. \tag{14}$$

Fix $h, \mathbf{x}$, and $\mathbf{y}$, and let $\mathbf{y}'$ achieve the margin in (12). Then, we get

$$M(h,\mathbf{x},\mathbf{y}) = h(\mathbf{x},\mathbf{y}) - h(\mathbf{x},\mathbf{y}'), \text{ and } M(f,\mathbf{x},\mathbf{y}) \leq f(\mathbf{x},\mathbf{y}) - f(\mathbf{x},\mathbf{y}').$$

With these two results, since $\langle f(\mathbf{x},\mathbf{y}) - f(\mathbf{x},\mathbf{y}') \rangle_{f\sim p^m} = h(\mathbf{x},\mathbf{y}) - h(\mathbf{x},\mathbf{y}')$, we can get

$$\begin{aligned}
\langle \Pr_Q[M(f,\mathbf{x},\mathbf{y}) > \frac{\gamma}{2} | M(h,\mathbf{x},\mathbf{y}) \leq 0] \rangle_{f\sim p^m} &= \langle \Pr_{f\sim p^m}[M(f,\mathbf{x},\mathbf{y}) > \frac{\gamma}{2} | M(h,\mathbf{x},\mathbf{y}) \leq 0] \rangle_Q \\
&\leq \langle \Pr_{f\sim p^m}[f(\mathbf{x},\mathbf{y}) - f(\mathbf{x},\mathbf{y}') > \frac{\gamma}{2} | M(h,\mathbf{x},\mathbf{y}) \leq 0] \rangle_Q \\
&\leq \langle \Pr_{f\sim p^m}[f(\mathbf{x},\mathbf{y}) - f(\mathbf{x},\mathbf{y}') - M(h,\mathbf{x},\mathbf{y}) > \frac{\gamma}{2}] \rangle_Q \\
&\leq e^{-\frac{m\gamma^2}{32c^2}},
\end{aligned} \tag{15}$$

where the first two inequalities are due to the fact that if two events $A \subseteq B$, then $p(A) \leq p(B)$, and the last inequality is due to the Hoeffding's inequality.

Substitute (15) into (14), and we get,

$$\Pr_Q[M(h,\mathbf{x},\mathbf{y}) \leq 0] \leq \langle \Pr_Q[M(f,\mathbf{x},\mathbf{y}) \leq \frac{\gamma}{2}] \rangle_{f\sim p^m} + e^{-\frac{m\gamma^2}{32c^2}}. \tag{16}$$

Let p_0^m be a prior distribution on $\mathcal{H}_m$. p_0^m is constructed from p_0 over $\mathcal{H}$ exactly as p^m is constructed from p. Then, $KL(p^m||p_0^m) = mKL(p||p_0)$. By the PAC-Bayes theorem (McAllester, 1999), with probability at least $1-\delta$ over sample $\mathcal{D}$, the following bound holds for any distribution p,

$$\langle \Pr_Q[M(f,\mathbf{x},\mathbf{y}) \leq \frac{\gamma}{2}] \rangle_{f\sim p^m} \leq \langle \Pr_{\mathcal{D}}[M(f,\mathbf{x},\mathbf{y}) \leq \frac{\gamma}{2}] \rangle_{f\sim p^m} + \sqrt{\frac{mKL(p||p_0) + \ln N + \ln \delta^{-1} + 2}{2N-1}}. \tag{17}$$

By the similar statement as in (13), for every $f \in \mathcal{H}_m$ we have,

$$\Pr_{\mathcal{D}}[M(f,\mathbf{x},\mathbf{y}) \leq \frac{\gamma}{2}] \leq \Pr_{\mathcal{D}}[M(h,\mathbf{x},\mathbf{y}) \leq \gamma] + \Pr_{\mathcal{D}}[M(f,\mathbf{x},\mathbf{y}) \leq \frac{\gamma}{2} | M(h,\mathbf{x},\mathbf{y}) > \gamma].$$

Taking the expectation at both sides w.r.t $f \sim p^m$, we can get

$$\begin{aligned}\langle \Pr_{\mathcal{D}}[M(f,\mathbf{x},\mathbf{y}) \leq \frac{\gamma}{2}] \rangle_{f\sim p^m} \quad &\leq \Pr_{\mathcal{D}}[M(h,\mathbf{x},\mathbf{y}) \leq \gamma] \\ &+ \langle \Pr_{\mathcal{D}}[M(f,\mathbf{x},\mathbf{y}) \leq \frac{\gamma}{2} | M(h,\mathbf{x},\mathbf{y}) > \gamma] \rangle_{f\sim p^m}, \end{aligned} \tag{18}$$

where the second term at the right hand side equals to $\langle \Pr_{f\sim p^m}[M(f,\mathbf{x},\mathbf{y}) \leq \frac{\gamma}{2} | M(h,\mathbf{x},\mathbf{y}) > \gamma] \rangle_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}$ by exchanging the orders of expectations, and we can get

$$\begin{aligned}\Pr_{f\sim p^m}\left[M(f,\mathbf{x},\mathbf{y}) \leq \frac{\gamma}{2} | M(h,\mathbf{x},\mathbf{y}) > \gamma\right] \quad &= \Pr_{f\sim p^m}\left[\exists \mathbf{y}' \neq \mathbf{y} : \Delta f(\mathbf{x},\mathbf{y}') \leq \frac{\gamma}{2} | \forall \mathbf{y}' \neq \mathrm{y} : \Delta h(\mathbf{x},\mathbf{y}') > \gamma\right] \\ &\leq \Pr_{f\sim p^m}\left[\exists \mathbf{y}' \neq \mathbf{y} : \Delta f(\mathbf{x},\mathbf{y}') \leq \frac{\gamma}{2} | \Delta h(\mathbf{x},\mathbf{y}') > \gamma\right] \\ &\leq \sum_{\mathbf{y}'\neq\mathbf{y}} \Pr_{f\sim p^m}\left[\Delta f(\mathbf{x},\mathbf{y}') \leq \frac{\gamma}{2} | \Delta h(\mathbf{x},\mathbf{y}') > \gamma\right] \\ &\leq (|\mathcal{Y}|-1) e^{-\frac{m\gamma^2}{32c^2}}, \end{aligned} \tag{19}$$

where we use $\Delta f(\mathbf{x},\mathbf{y}')$ to denote $f(\mathbf{x},\mathbf{y}) - f(\mathbf{x},\mathbf{y}')$, and use $\Delta h(\mathbf{x},\mathbf{y}')$ to denote $h(\mathbf{x},\mathbf{y}) - h(\mathbf{x},\mathbf{y}')$.

Put (16), (17), (18), and (19) together, then we get following bound holding for any fixed m and $\gamma > 0$,

$$\Pr_Q[M(h,\mathbf{x},\mathbf{y}) \leq 0] \quad \leq \Pr_{\mathcal{D}}[M(h,\mathbf{x},\mathbf{y}) \leq \gamma] + |\mathcal{Y}| e^{-\frac{m\gamma^2}{32c^2}} + \sqrt{\frac{mKL(p||p_0) + \ln N + \ln \delta^{-1} + 2}{2N-1}}.$$

To finish the proof, we need to remove the dependence on m and γ. This can be done by applying the union bound. By the definition of f, it is obvious that if $f \in \mathcal{H}_m$ then $f(\mathbf{x},\mathbf{y}) \in \{(2k-m)c/m : k = 0,1,\ldots,m\}$. Thus, even though γ can be any positive value, there are no more than $m+1$ events of the form $\{M(f,\mathbf{x},\mathbf{y}) \leq \gamma/2\}$. Since only the application of PAC-Bayes theorem in (17) depends on (m,γ) and all the other steps are true with probability one, we just need to consider the union of countably many events. Let $\delta_{m,k} = \delta/(m(m+1)^2)$, then the union of all the possible events has a probability at most $\sum_{m,k}\delta_{m,k} = \sum_m (m+1)\delta/(m(m+1)^2) = \delta$. Therefore, with probability at least $1-\delta$ over random samples of $\mathcal{D}$, the following bound holds for all m and all $\gamma > 0$,

$$\begin{aligned}\Pr_Q[M(h,\mathbf{x},\mathbf{y}) \leq 0] - \Pr_{\mathcal{D}}[M(h,\mathbf{x},\mathbf{y}) \leq \gamma] \quad &\leq |\mathcal{Y}| e^{-\frac{m\gamma^2}{32c^2}} + \sqrt{\frac{mKL(p||p_0) + \ln N + \ln \delta_{m,k}^{-1} + 2}{2N-1}} \\ &\leq |\mathcal{Y}| e^{-\frac{m\gamma^2}{32c^2}} + \sqrt{\frac{mKL(p||p_0) + \ln N + 3\ln\frac{m+1}{\delta} + 2}{2N-1}}. \end{aligned}$$

Setting $m = \lceil 16c^2\gamma^{-2} \ln \frac{N|\mathcal{Y}|^2}{KL(p||p_0)+1} \rceil$ gives the results in the theorem. ■

References

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden Markov support vector machines. In *International Conference on Machine Learning*, 2003.

Yasemin Altun, David McAllester, and Mikhail Belkin. Maximum margin semi-supervised learning for structured variables. In *Advances in Neural Information Processing Systems*, 2006.

Galen Andrew and Jianfeng Gao. Scalable training of l_1-regularized log-linear models. In *International Conference on Machine Learning*, 2007.

David F. Andrews and Colin L. Mallows. Scale mixtures of normal distributions. *Journal of the Royal Statistical Society*, B(1):99–102, 1974.

Peter L. Bartlett, Michael Collins, Ben Taskar, and David McAllester. Exponentiated gradient algorithms for larg-margin structured classification. In *Advances in Neural Information Processing Systems*, 2004.

Kristin P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optim. Methods Softw*, (1):23–34, 1992.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Ulf Brefeld and Tobias Scheffer. Semi-supervised learning for structured output variables. In *International Conference on Machine Learning*, 2006.

Antoni B. Chan, Nuno Vasconcelos, and Gert R. G. Lanckriet. Direct convex relaxations of sparse SVM. In *International Conference on Machine Learning*, 2007.

Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *International Conference on Machine Learning*, 2008.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projection onto the l_1-ball for learning in high dimensions. In *International Conference on Machine Learning*, 2008.

Mirosla Dudík, Steven J. Phillips, and Robert E. Schapire. Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, (8):1217–1260, 2007.

Susana Eyheramendy, Alexander Genkin, Wen-Hua Ju, David D. Lewis, and David Madiagan. Sparse Bayesian classifiers for text categorization. Technical report, Rutgers University, 2003.

Mario Figueiredo. Adaptive sparseness for supervised learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(9):1150–1159, 2003.

Mario A. T. Figueiredo. Adaptive sparseness using Jeffreys prior. In *Advances in Neural Information Processing Systems*, 2001.

Amir Globerson, Terry Y. Koo, Xavier Carreras, and Michael Collins. Exponentiated gradient algorithms for log-linear structured prediction. In *International Conference on Machine Learning*, 2007.

Tommi Jaakkola, Marina Meila, and Tony Jebara. Maximum entropy discrimination. In *Advances in Neural Information Processing Systems*, 1999.

Tony Jebara. Multitask sparsity via maximum entropy discrimination. *To appear in Journal of Machine Learning Research*, 2009.

Tony Jebara and Tommi Jaakkola. Feature selection and dualities in maximum entropy discrimination. In *Uncertainty in Artificial Intelligence*, 2000.

Ata Kaban. On Bayesian classification with laplace priors. *Pattern Recognition Letters*, 28(10): 1271–1282, 2007.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

John Langford and John Shawe-Taylor. Pac-Bayes & margins. In *Advances in Neural Information Processing Systems*, 2003.

John Langford, Matthias Seeger, and Nimrod Megiddo. An improved predictive accuracy bound for averaging classifiers. In *International Conference on Machine Learning*, 2001.

Guy Lebanon and John Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems*, 2001.

Su-In Lee, Varun Ganapathi, and Daphne Koller. Efficient structure learning of Markov networks using l_1-regularization. In *Advances in Neural Information Processing Systems*, 2006.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, (45):503–528, 1989.

David McAllester. Generalization bounds and consistency for structured labeling. In *Predicting Structured Data, edited by G. Bakir, T. Hofmann, B. Scholkopf, A. Smola, B. Taskar, and S. V. N. Vishwanathan. MIT Press*, 2007.

David McAllester. PAC-Bayesian model averaging. In *the Twelfth Annual Conference on Computational Learning Theory*, 1999.

Yuan (Alan) Qi, Martin Szummer, and Thomas P. Minka. Bayesian conditional random fields. In *International Conference on Artificial Intelligence and Statistics*, 2005.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems*, 2004.

Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, (77(2)):257–286, 1989.

Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. (online) subgradient methods for structured prediction. In *International Conference on Artificial Intelligence and Statistics*, 2007.

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, 1998.

Peter Sollich. Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Journal of Machine Learning Research*, (3):21–52, 2002.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, 2003.

Ben Taskar, Simon Lacoste-Julien, and Michael I. Jordan. Structured prediction via the extragradient method. In *Advances in Neural Information Processing Systems*, 2006.

Robert Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society*, B(58):267–288, 1996.

Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, (1):211–244, 2001.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*, 2004.

Martin J. Wainwright, Pradeep Ravikumar, and John Lafferty. High-dimensional graphical model selection using l_1-regularized logistic regression. In *Advances in Neural Information Processing Systems*, 2006.

David Wipf, Jason Palmer, and Bhaskar Rao. Perspectives on sparse bayesian learning. In *Advances in Neural Information Processing Systems*, 2003.

Linli Xu, Dana Wilkinson, Finnegan Southey, and Dale Schuurmans. Discriminative unsupervised learning of structured predictors. In *International Conference on Machine Learning*, 2006.

Zhihua Zhang and Michael I. Jordan. Bayesian multicategory support vector machines. In *Uncertainty in Artificial Intelligence*, 2006.

Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-norm support vector machines. In *Advances in Neural Information Processing Systems*, 2004.

Jun Zhu and Eric P. Xing. On primal and dual sparsity of Markov networks. In *International Conference on Machine Learning*, 2009.

Jun Zhu, Zaiqing Nie, Bo Zhang, and Ji-Rong Wen. Dynamic hierarchical Markov random fields for integrated web data extraction. *Journal of Machine Learning Research*, (9):1583–1614, 2008a.

Jun Zhu, Eric P. Xing, and Bo Zhang. Laplace maximum margin Markov networks. In *International Conference on Machine Learning*, 2008b.

Jun Zhu, Eric P. Xing, and Bo Zhang. Partially observed maximum entropy discrimination Markov networks. In *Advances in Neural Information Processing Systems*, 2008c.

Jun Zhu, Amr Ahmed, and Eric P. Xing. MedLDA: Maximum margin supervised topic models for regression and classification. In *International Conference on Machine Learning*, 2009a.

Jun Zhu, Eric P. Xing, and Bo Zhang. Primal sparse max-margin Markov networks. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2009b.

Journal of Machine Learning Research 10 (2009) 2571-2613 Submitted 12/07; Revised 1/09; Published 11/09

Learning When Concepts Abound

Omid Madani MADANI@AI.SRI.COM
SRI International, AI Center
333 Ravenswood Ave
Menlo Park, CA 94025

Michael Connor CONNOR2@UIUC.EDU
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Wiley Greiner W.GREINER@LASOFT.COM
Los Angeles Software Inc
Santa Monica, CA 90405

Editor: Ralf Herbrich

Abstract

Many learning tasks, such as large-scale text categorization and word prediction, can benefit from efficient training and classification when the number of classes, in addition to instances and features, is large, that is, in the thousands and beyond. We investigate the learning of sparse class *indices* to address this challenge. An index is a mapping from features to classes. We compare the index-learning methods against other techniques, including one-versus-rest and top-down classification using perceptrons and support vector machines. We find that index learning is highly advantageous for space and time efficiency, at both training and classification times. Moreover, this approach yields similar and at times better accuracies. On problems with hundreds of thousands of instances and thousands of classes, the index is learned in minutes, while other methods can take hours or days. As we explain, the design of the learning update enables conveniently constraining each feature to connect to a small subset of the classes in the index. This constraint is crucial for scalability. Given an instance with l active (positive-valued) features, each feature on average connecting to d classes in the index (in the order of 10s in our experiments), update and classification take $O(dl \log(dl))$.

Keywords: index learning, many-class learning, multiclass learning, online learning, text categorization

1. Introduction

A fundamental activity of intelligence is to repeatedly and rapidly categorize. Categorization (classification or prediction) has a number of uses; in particular, categorization enables inferences and the taking of appropriate actions in different situations. Advanced intelligence, whether of animals or artificial systems, may require effectively working with myriad classes (concepts or categories). How can a system quickly classify when the number of classes is huge (Figure 1), that is, in the thousands and beyond? In nature, this problem of rapid classification in the presence of many classes may have been addressed (for evidence of fast classification in the visual domain, see Thorpe et al. 1996 and Grill-Spector and Kanwisher 2005). Furthermore, ideally, we seek systems that *efficiently*

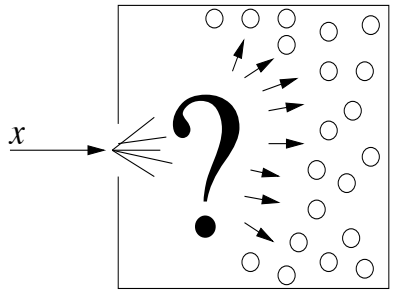


Figure 1: The problem of quick classification in the presence of myriad classes: How can a system quickly classify a given instance, specified by a feature vector $x \in R^n$, into a small subset of classes from among possibly millions of candidate classes (shown by small circles)? How can a system *efficiently learn* to quickly classify?

learn to efficiently classify in the presence of myriad classes. Many tasks can be viewed as instantiations of this *large-scale many-class* learning problem, including: (1) classifying text fragments (such as queries, advertisements, news articles, or web pages) into a large collection of categories, such as the ones in the Yahoo! topic hierarchy (`http://dir.yahoo.com`) or the Open Directory Project (`http://dmoz.org`) (e.g., Dumais and Chen, 2000; Liu et al., 2005; Madani et al., 2007; Xue et al., 2008), (2) statistical language modeling and similar prediction problems (e.g., Goodman, 2001; Even-Zohar and Roth, 2000; Madani et al., 2009), and (3) determining the visual categories for image tagging, object recognition, and multimedia retrieval (e.g., Wang et al., 2001; Forsyth and Ponce, 2003; Fidler and Leonardis, 2007; Chua et al., 2009; Aradhye et al., 2009). The following realization is important: in many prediction tasks, such as predicting words in text (statistical language modeling), training data is abundant because the class labels are not costly, that is, *the source of class feedback (the labels) need not be explicit assignment by humans* (see also Section 3.1).

To classify an instance, applying binary classifiers, one by one, to determine the correct class(es) is quickly rendered impractical with increasing number of classes. Moreover, learning binary classifiers can be too costly with large numbers of classes and instances (millions and beyond). Other techniques, such as nearest neighbors, can suffer similar drawbacks, such as prohibitive space requirements, possibly slow classification speeds, or poor generalization. Ideally, we desire scalable discriminative learning methods that learn compact classification systems that attain adequate accuracy.

One idea for achieving quick classification is to use the features of the given instance as cues to dramatically reduce the space of possibilities, that is, to build and update a mapping, or an *index*, from features to the classes. We explore this idea in this work. An index here is a weighted bipartite graph that connects each feature to zero or more classes. During classification, given an instance containing certain features, the index is used ("looked up") much like a typical inverted index for document retrieval would be. Here, classes are retrieved and ranked by the scores that they obtain during retrieval, as we describe. The ranking or the scores can then be used for class assignment. In this work, we explore the learning of such cues and connections, which we refer to as *index learning*. For this approach to be effective overall, roughly, two properties need to hold: To achieve adequate accuracy and efficiency and in many problems arising in practice, (1) each feature need only connect

to a relatively small number of classes, and (2) these connections can be discovered efficiently. We provide empirical evidence for these conjectures by presenting efficient and competitive indexing algorithms.

We design our algorithms to efficiently learn sparse indices that yield accurate class rankings. As we explain, the computations may best be viewed as being carried out from the side of features. During learning, each feature determines to which relatively few classes it should lend its weights (votes) to, subject to (space) efficiency constraints. This parsimony in connections is achieved by a kind of *sparsity-preserving* updates. Given an instance with l active (i.e., positive-valued) features, each feature on average connecting to d classes in the index, update and classification take $O(dl\log(dl))$ operations. d is in the order of 10s in our experiments. The approach we develop uses ideas from online learning and multiclass learning, including mistake driven and margin-based updates, and expert aggregation (e.g., (e.g., Rosenblatt, 1958; Genest and Zidek, 1986; Littlestone, 1988; Crammer and Singer, 2003a), as well as the idea of the inverted index, a core data structure in information retrieval (e.g., Witten et al., 1994; Turtle and Flood, 1995; Baeza-Yates and Ribeiro-Neto, 1999).

We empirically compare our algorithms to one-versus-rest and top-down classifier based methods (e.g., Rifkin and Klautau, 2004; Liu et al., 2005; Dumais and Chen, 2000), and to the first proposal for index learning by Madani et al. (2007). We use linear classifiers—perceptrons and support vector machines—in the one-versus-rest and top-down methods. One-versus-rest is a simple strategy that has been shown to be quite competitive in accuracy in multiclass settings, when properly regularized binary classifiers are used (Rifkin and Klautau, 2004), and linear support vector machines achieve the state of the art in accuracy in many text classification problems (e.g., Sebastiani, 2002; Lewis et al., 2004). Hierarchical training and classification is a fairly scalable and conceptually simple method that has commonly been used for large-scale text categorization (e.g., Koller and Sahami, 1997; Dumais and Chen, 2000; Dekel et al., 2003; Liu et al., 2005).

In our experiments on six text categorization data sets and one word prediction problem, we find that the index is learned in seconds or minutes, while the other methods can take hours or days. The index learned is more efficient in its use of space than those of the other classification systems, and yields quicker classification time. Very importantly, we find that budgeting the connections of the features is a major factor in rendering the approach scalable. We explain how the design of the update makes this budget enforcement convenient. We have observed that the accuracies are as good as and at times better than the best of the other methods that we tested. As we explain, methods based on binary classifiers, such as one-versus-rest and top-down, are at a disadvantage in our many-class tasks, not just in terms of efficiency but also in accuracy. The indexing approach is simple: it requires neither taxonomies, nor extra feature reduction preprocessing. Thus, we believe that index learning offers a viable option for various many-class settings.

The contribution of this paper include:

- Raising the problem of large-scale many-class learning, with the goal of achieving both efficient classification and efficient training
- Proposing and exploring index learning, and developing a novel weight-update method in the process
- Empirically comparing index learning to several commonly used techniques, on a range of small and large problems and under several evaluation measures of accuracy and space and

time efficiency, and providing evidence that very scalable systems are possible without sacrificing accuracy

This paper is organized as follows. In Section 2, we discuss related work. In Section 3, we describe and motivate the learning problem, independent of the solution strategy. We explain the index, and describe our implementation and measures of index quality, in terms of both accuracy and efficiency. We then report on the NP-hardness of a formalization of the index learning problem. In Section 4, we present our index learning approach. Throughout this section, we discuss and motivate the choices in the design of the algorithms. In particular, the consideration of what each feature should do in isolation turns out to be very useful. In Section 5, we briefly describe the other methods we compare against, including the one-versus-rest and top-down methods. In Section 6, we present a variety of experiments. We report on comparisons among the techniques and our observations on the effects of parameter choices and tradeoffs. In Section 7, we summarize and provide concluding thoughts. In the appendices, we present a proof of NP-hardness and additional experiments.

2. Related Work

Related work includes multiclass learning and online learning, expert methods, indexing, streaming algorithms, and concepts in cognitive psychology.

There exists much work on multiclass learning, including nearest neighbors approaches, naive Bayes, support vector machine variants, one-versus-rest and output-codes (see, for example, Hastie et al., 2001; Rennie et al., 2003; Dietterich and Bakiri, 1995); however, the focus has not been scalability to very large numbers of classes.

Multiclass online algorithms with the goal of obtaining good rankings include the multiclass and multilabel perceptron (MMP) algorithm (Crammer and Singer, 2003a) and subsequent work (e.g., Crammer and Singer, 2003b; Crammer et al., 2006). These algorithms are very flexible and include both additive and multiplicative variants, and may optimize an objective in each update; some variants can incorporate non-linear kernel techniques. We may refer to them as prototype methods because the operations (such as weight adjustments and imposing various constraints) can be viewed as being performed on the (prototype) weight vector for each class. In our indexing algorithms it is the features that update and normalize their connections to the classes. This difference is motivated by efficiency (for further details, see Sections 4.1 and 4.4, and the experiments). Similar to the perceptron algorithm (Rosenblatt, 1958), we use a variant of mistake driven updating. The variant is based on trying to achieve and preserve a margin during online updating. Learning to improve or optimize some measure of margin has been shown to improve generalization (Vapnik, 2000). On use of margin for online methods, see for instance Krauth and Mezard (1987), Anlauf and Biehl (1989), Freund and Schapire (1999), Gentile (2001), Li and Long (2002), Li et al. (2002), Crammer et al. (2006) and Carvalho and Cohen (2006). In our setting, a simple example shows that keeping a margin can be beneficial over pure mistake-driven updating even when considering a single feature in isolation (Section 4.3.1).

The indexing approach in its focus on features (predictors) has similarities with additive models and tree-induction algorithms (Hastie et al., 2001), and may be viewed as a variant of so-called expert (opinion or forecast) aggregation and weight learning (e.g., Mesterharm, 2000; Freund et al., 1997; Cesa-Bianchi et al., 1997; Vovk, 1990; Genest and Zidek, 1986). In the standard experts problems, all or most experts provide their output, and the output is usually binary or a probability

(the outcome to predict is binary). In our setting, a relatively small set of features are active in each instance, and only those features are used for voting and ranking. In this respect, the problem is in the setting of the "sleeping" or "specialist" experts scenarios (Freund et al., 1997; Cohen and Singer, 1999). Differences or special properties of our setting include the fact that here each expert provides a partial class-ranking with its votes, the votes can change over time (not fixed), and the pattern of change is dependent on the algorithm used (the experts are not "autonomous"). In a multiclass calendar scheduling task (Blum, 1997), Blum investigates an algorithm in which each feature votes for (connects to) the majority class in the past 5 classes seen for that feature (the classes of the most recent 5 instances in which the feature was active). This design choice was due to the temporal (drifting) nature of the learning task. Feature weights for the goodness of the features are learned (in a multiplicative or Winnow style manner). Mesterharm refers to such features (or experts) as subexperts (Mesterharm, 2000, 2001), as the performance can be significantly enhanced by learning a good weighting for mixing (aggregating) the experts' votes,[1] and it is shown how different linear threshold algorithms can be extended to the multiclass weight learning setting. The classifier is referred to as a *linear-max* classifier, since the maximum scoring class is assigned to the instance (as opposed to a linear-threshold classifier). Mesterharm's work includes the case where the experts may cast probabilities for each class, but the focus is not on how the features may compute such probabilities (it is assumed the experts are given). Learning different weights for the features can complement indexing techniques. Section 4.3.2 gives a limited form of differential expert weighting (see also Madani, 2007a).

The one-versus-rest technique (e.g., Rifkin and Klautau, 2004) and use of a class hierarchy (taxonomy) (e.g., Liu et al., 2005; Dumais and Chen, 2000; Koller and Sahami, 1997) for top-down training are simple intuitive techniques commonly used for text categorization. The use of the structure of a taxonomy for training and classification offers a number of efficiency and/or accuracy advantages (Koller and Sahami, 1997; Liu et al., 2005; Dumais and Chen, 2000; Dekel et al., 2003; Xue et al., 2008), but also can present several drawbacks. Issues such as multiple taxonomies, evolving taxonomies, unnecessary intermediate categories on the path from the root to deeper categories, or unavailability of a taxonomy are all difficulties for the tree-based approaches. In our experiments, we find that index learning offers both several efficiency advantages and ease of use (Section 6). No taxonomy or separate feature-reduction pre-processing is required. Indeed, our method can be viewed as a feature selection or reduction method. On the other hand, researchers have shown some accuracy advantages from the use of the taxonomy structure (e.g., top-down) compared to "flat" one-versus-rest training (in addition to efficiency) (e.g., Dumais and Chen, 2000; Liu et al., 2005; Dekel et al., 2003) (this depends somewhat on the particular method and the loss used). Our current indexing approach is flat (but see Huang et al. 2008, for a two-stage nonlinear method using fast index learning for the first stage). One advantage that classifier-based methods such as one-versus-rest and top-down may offer is that the training can be highly parallelized: learning of each binary classifier can be carried out independent of the others.

The inverted index, for instance from terms to documents, is a fundamental data structure in information retrieval (Witten et al., 1994; Baeza-Yates and Ribeiro-Neto, 1999). Akin to the TFIDF weight representation and variants, the index learned is also weighted. However, in our case, the classes (to be indexed), unlike the documents, are implicit, indirectly specified by the training instances (the instances are not the "documents" to be indexed), and the index construction becomes

1. Theoretical work often focuses the analysis on learning the best expert, and the use of term "subexpert" is introduced by Mesterharm to differentiate.

a learning problem. As one simple consequence, the presence of a feature in a training instance that belongs to class c does not imply that the feature will point to class c in the index learned. We give a baseline algorithm, similar to TFIDF index construction in its independent computation of weights, in Section 4.2. Indexing has also been used to speed up nearest neighbor methods, classification, and retrieval and matching schemes (e.g., Grobelnik and Mladenic, 1998; Bayardo et al., 2007; Fidler and Leonardis, 2007). Indexing could be used to index already trained (say linear) classifiers, but the issues of space and time efficient learning remain, and accuracy can suffer when using binary classifiers for class ranking (see Section 6.1). Learning of an unweighted index was introduced by Madani et al. (2007), in which the problem of efficient classification under myriad classes was motivated. This two-stage approach is explained in Section 6.4.1, and we see in Section 6.4.1 that learning a weighted index to improve ranking appears to be a better strategy than the original approach in terms of accuracy, as well as simplicity and efficiency. Subsequent work on indexing by Madani and Huang (2008) explores further variations and advances to feature updating (e.g., supporting nonstationarity and hinge-loss minimization), taking as a starting point the findings of this work on the benefits of efficient feature updating. It also includes comparisons with additional multiclass approaches. This paper is an extension of the work by Madani and Connor (2008).

The field of data-streaming algorithms studies methods for efficiently computing statistics of interest over data streams, for example, reporting the items with proportions exceeding a threshold, or the highest k proportion items (sometimes called "hot-list" or "iceberg" queries). This is to be achieved under certain efficiency constraints, for example, with at most two passes over the data and poly logarithmic space (e.g., see Fang et al., 1998; Gibbons and Matias, 1999). Note that in the case of a single feature, if we only value good rankings, computing weights may not be necessary, but in the general case of multiple features, the weights become the votes given to each class, and are essential in significantly improving the final rankings. An algorithm similar to our single-feature update for the Boolean case is used as a subroutine by Karp et al. (2003), for efficiently computing most frequent items. In some scenarios, drifts in proportions can exist, and then online and possibly competitive measures of performance may become important (Borodin and El Yaniv, 1998; Albers and Westbrook, 1998). In this ranking and drifting respect, the feature-update task has similarities with online list-serving and caching (Borodin and El Yaniv, 1998), although we may assume that the sequence is randomly ordered (at minimum, not ordered by an adversary). Some connections and differences between goals in machine learning research and space-efficient streaming and online computations are discussed by Guha and McGregor (2007).

Statistical language modeling and similar prediction tasks are often accomplished by n-gram (Markov) models (Goodman, 2001), but the supervised (or discriminative) approach may provide superior performance due to its potential for effectively aggregating richer feature sets (Even-Zohar and Roth, 2000; Madani et al., 2009). Prior work has focused on discriminating within a small (confusion) set of possibilities. In the related task of *prediction games* (Madani, 2007a,b), Madani proposes and explores an integrated learning activity in which a system builds its own classes to be predicted and to help predict. That approach involves large-scale long-term online learning, where the number of concepts grows over time, and can exceed millions.

Concepts and various phenomena associated with them have been studied extensively in cognitive psychology (e.g., Murphy, 2002). A general question that motivated our work, and that appears heretofore uninvestigated, is the question of computational processes required for a system to effectively deal with a huge number of concepts. Three prominent theories on the nature of the representation of concepts are the classical theory (logical representations), the exemplar theory (akin

to nearest neighbors), and the prototype theory (akin to linear feature-based representations). Prototype theory is perhaps the most successful in explaining various observed phenomena regarding human concepts (Murphy, 2002). Interestingly, our work suggests a predictor-based representation for efficient recall/recognition purposes, that is, the representation of a concept, at a minimum for recall/retrieval purposes, is distributed among the features (predictors or cues). However, the predictor-based representation remains closest to the prototype theory.

3. Many-Class Learning and Indexing

In this section, we first present the learning setting and introduce some notation in the process. Next, we motivate many-class learning and the indexing approach. In Section 3.2, we define the index and how it is implemented and used in this work. We then present our accuracy and efficiency evaluation measures in Section 3.3. Before moving to index learning (Section 4), we analyze the computational complexity of a formulation of index learning in Section 3.4.

A learning problem consists of a collection S of instances, where S can denote a finite set, or, in the online setting, a sequence of instances. Each training instance is specified by a vector of feature values, $\mathbf{v_x}$, as well as a class (or assigned label) that the instance belongs to,[2] c_x. Thus each instance x is a pair $\langle \mathbf{v_x}, c_x \rangle$. $\mathcal{F}$ and $\mathcal{C}$ denote respectively the set of all features and classes. Our proposed algorithms ignore features with nonpositive value,[3] and in our experiments feature values range in $[0,1]$. $\mathbf{v_x}[f]$ denotes the value of feature f in the vector of features of instance x, where $\mathbf{v_x}[f] \geq 0$. If $\mathbf{v_x}[f] > 0$, we say feature f is *active* (in instance x), and denote this aspect by $f \in x$. Thus, an instance may be viewed as a set of active features, and the input problem may be seen as a tripartite graph (Figure 2). The number of active features is denoted by $|x|$. We also use the expression $x \in c$ to denote that instance x belongs to class c (c is a class of x).

As an example, in text categorization, a "document" (e.g., an email, an advertisement, a news article, etc.) is typically translated to a vector by a "bag of words" method as follows. Each term (e.g., "ball", "cat", "the", "victory", ...) is assigned an exclusive unique integer id. The finite set of words (more generally phrases or ngrams), those appearing in at least one document in the data set, comprise the set of features $\mathcal{F}$. Thus the vector $\mathbf{v_x}$ corresponding to a document lives in an $|\mathcal{F}|$ dimensional space, where $\mathbf{v_x}[i] = k$ iff the word with id i (corresponding to dimension i) appears k times in the document, where $k \geq 0$ (other possibilities for feature values include Boolean and TFIDF weighting). Therefore, in typical text categorization tasks, the number of active features in an instance is the number of unique words that appear in the corresponding document. The documents in the training set are assigned zero or more true class ids as well. Section 6 describes further the feature representation and other aspects of our experimental data. For background on machine learning in particular when applied to text classification, please refer to Sebastiani (2002) or Lewis et al. (2004).

2. In this paper, to keep the treatment focused, and for simplicity of evaluation and algorithm description, we treat the multiclass but single class (label) per instance setting. However, two of our seven data sets include multilabel instances. Whenever necessary, we briefly note the changes needed, for example, to the algorithm, to handle multiple labels. However, the multilabel setting may require additional treatment for better accuracy.
3. A partial remedy is to replace each feature that can also have negative values by two features, one having value $\max(v,0)$, the other $\max(0,-v)$.

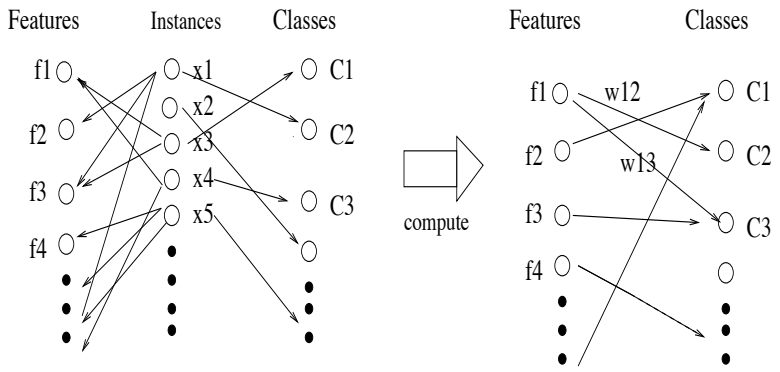


Figure 2: A depiction of the problem: the input can be viewed as a tripartite graph, possibly weighted, and perhaps only seen one instance at a time in an online manner. Our goal is to learn an accurate efficient index, that is, a sparse weighted bipartite graph that connects each feature to zero or more classes, such that an adequate level of accuracy is achieved when the index is used for classification. The instances are ephemeral: they serve only as intermediaries in affecting the connections from features to classes. The index to learn is also equivalent to a sparse weight matrix (in which the entries are nonnegative in our current work) (see Sections 3.2 and 3.2.1).

3.1 The Level of Human Involvement in Teaching and Many-Class Learning

Learning under myriad-classes is not confined to a few text-classification problems. There are a number of tasks that could be viewed as problems with many classes and, if effective many-class methods are developed, such an interpretation can be quite useful. In terms of the sources of the classes, we may roughly distinguish supervised learning problems along the following dimensions (the roles of the teacher):

1. The source that defines the classes of interest, that is, the space of the target classes to predict.

2. The source of supervisory feedback, that is, the source or the process that assigns to each instance one or more class labels, using the defined set of classes. This is necessary for the procurement of training data, for supervised learning.

In many familiar cases, the classes are both human-defined and human-assigned. These include typical text classification problems (e.g., see Lewis et al. 2004 and Open Directory Project or Yahoo! directories/topics). In many others, class assignment is achieved by some "natural" or indirect activity, that is, the "labeling" process is not as explicit or controlled. The labeling is a by-product of an activity carried out for other purposes. One example of this case is data sets obtained from news groups postings (e.g., Lang, 1995). In this case, users post or reply to messages, without necessarily verifying whether their message is topically relevant to the group. Another example problem is predicting words using the context that the word appears (the words are the classes). In these problems, the set of the classes of interest may be viewed as human-defined, but the labeling is implicit (collections of written or spoken texts in the word prediction task). The extreme case

where both the set of classes and the labeling is achieved with little or no human involvement is also possible, and we believe very important. For instance, Madani (2007b,a) explores tasks in which it is (primarily) the machine that builds its own many concepts, through experience, and learns prediction connections among them. This is a kind of *autonomous* learning. As human involvement and control diminishes over the learning process, the amount of noise tends to increase. However, training data as well as the number of classes can increase significantly. We have used the term "many-class" (in contrast to multiclass) to emphasize this aspect of the large number of classes in these problems.

Thus, in large-scale many-class learning problems, all the three sets S, $\mathcal{C}$, and $\mathcal{F}$ can be huge. For instance, in experiments reported here, $\mathcal{C}$ and $\mathcal{F}$ can be in the tens of thousands, and S can be in the millions. S can be an infinite stream of instances and $\mathcal{C}$ and $\mathcal{F}$ can grow indefinitely in some tasks (e.g., Madani, 2007a). While $\mathcal{F}$ can be large (e.g., hundreds of thousands), in many applications such as text classification, instances tend to be relatively sparse: relatively a few of the features (tens or hundreds) are active in each instance.

The number of classes is so large that indexing them, not unlike the inverted index used for retrieval of documents and other object types, is a plausible approach. An important difference from traditional indexing is that classes, unlike documents, are implicit, specified only by the instances that belong to them. An index is a common technique for fast retrieval and classification, for instance to speed up nearest neighbor or nearest centroid computations (e.g., Grobelnik and Mladenic, 1998; Bayardo et al., 2007; Gabrilovich and Markovitch, 2007; Fidler and Leonardis, 2007). Also, for fast classification when there is a large number of classes, after one-versus-rest training of linear binary classifiers (see Section 5 on one-versus-rest training), a natural and perhaps necessary technique is to index the weights, that is, to build an index mapping each feature to those classifiers in which the feature has nonzero weight. This approach is indirect and does not adequately address efficient classification and space efficiency,[4] and the problem of slow training time for one-versus-rest training remains. Here, we propose to learn the index edges as well as their weights directly. For good classification performance as well as efficiency, we need to be very selective in the choice of the index entries, that is, which connections to create and with what weights. Figure 3 presents the basic cycle of categorization via index look up and learning via index updating (adjustments to connection weights). We have termed the system that is learned a *Recall System* (Madani et al., 2007): a system that, when presented with an instance, quickly "recalls" the appropriate classes from a potentially huge number of possibilities.

3.2 Index Definition, Implementation, and Use

The use of the index for retrieval, scoring, and ranking (classification) is similar to the use of inverted indexes for document retrieval. Here, features "index" classes instead of documents. In our implementation, for each feature there corresponds exactly one list that contains information about the feature's connections (similar to inverted or posting lists Witten et al. 1994 and Baeza-Yates and Ribeiro-Neto 1999). The list may be empty. Each entry in the list corresponds to a class that the feature is *connected* to. An entry in the list for feature f contains the id of a class c, as well as the *connection or edge weight* $w_{f,c}$, $w_{f,c} > 0$. Each class has at most one entry in a feature's list. If a class c doesn't have an entry in the list for feature f, then $w_{f,c}$ is taken to be 0. The connection

4. Our experiments show that if we do not drop some of the connections during learning, training and classification time and space consumption suffer significantly.

Basic Mode of Operation:
Repeat
1. Get next instance x
2. Retrieve, score, and rank classes via active features of x
3. If update condition is met:
3.1 Update index.
4. Zero (reset) the scores of the retrieved classes.
(a)

Algorithm RankedRetrieval(x**,** d_{max}**)**
/* initially, for each class c**, its score** s_c **is zero */**
1. For each active feature f **(i.e.,** $\mathbf{v_x}[f] > 0$**):**
For the first d_{max} **classes with highest connection weight to** f**:**
1.1. $s_c \leftarrow s_c + (r_f \times w_{f,c} \times \mathbf{v_x}[f])$
2. Return those classes with nonzero score, ranked by score.
(b)

Figure 3: (a) The cycle of classification and learning (updating). During pure classification (e.g., when testing), step 3 is skipped. See part (b) and Section 3.2 for how to use the index, and Section 4 for when and how to update the index. (b) The algorithm that uses a weighted index for retrieving and scoring classes. See Section 3.2.

weights are updated during learning. Our index learning algorithms keep the lists small for space and time efficiency (as we explain in Section 4.1). For ease of updating and efficiency, the lists are doubly linked circular dynamic lists in our implementation, and are kept sorted by weight.

Figure 3(b) shows how the index is used, via a procedure that we name RankedRetrieval. On presentation of an instance, the active features score the classes that they are connected to. The score that a class c receives, s_c, can be written as

$$s_c = \sum_{f \in x} r_f \times w_{f,c} \times \mathbf{v_x}[f], \tag{1}$$

where r_f is a measure of the predictiveness power or the *rating* of feature f, and we describe a method for computing it in Section 4.3.2. Currently, for simplicity, we may assume the rating is 1 for all features.[5] Note that the sum need only go over the entries in the list for each active feature (other weights are zero). We use a hash map to efficiently update the class scores during scoring. In a sense, each active feature casts votes for a subset of the classes, and those classes receive and tally their incoming votes (scores). In this work, the scores of the retrieved classes are positive. The positive scoring classes can then be ranked by their score, or, if it suffices, only the maximum scoring class can be kept track of and reported. Note that if negative scores (or edge weights) were allowed, then, when some true class obtains a negative score, the system would potentially have to process (i.e., retrieve or update) all zero scoring classes as well, hurting efficiency (this depends on how update and classification are defined). The scores of the retrieved classes are reset to 0 before the next call to RankedRetrieval.

On instance x, and with d connections per feature in the index, there can be at most $|\mathbf{v_x}|d$ unique classes scored. The average computation time of RankedRetrieval is thus $O(d|\mathbf{v_x}|\log(d|\mathbf{v_x}|))$, where d denotes the average number of connections of a randomly picked feature (from a randomly picked instance). In our implementation, for each active feature, only at most the d_{max} classes (25 in our experiments) with highest connection weights to the feature participate in scoring.

5. After index learning, r_f can be incorporated into the connection weights $w_{f,c}$.

3.2.1 GRAPH AND LINEAR-ALGEBRAIC VIEWS OF THE INDEX

A useful way of viewing the index is as a directed weighted bipartite graph (Figure 2): on one side there are features (one node per feature) and on the other side there are the classes. The index maps (connects) each feature to a subset of zero or more classes. An edge connecting feature f to class c has a positive weight denoted by $w_{f,c}$, or $w_{i,j}$ for feature i and class j, and corresponds to a list entry in the list for feature f. Absent edges have zero weight. The *outdegree* of a feature is the number of (outgoing) edges of the feature. Small feature outdegrees translates to efficiency in retrieval (and updating as we will see).

In addition to the graph-theoretic view, the index can also be seen as a sparse non-negative (weight) matrix $\mathbf{W}$. Let the rows correspond to the features and let the columns correspond to the classes. Retrieval or classification involves efficiently computing[6] the vector of class scores $\mathbf{v_x^T W}$, and post-processing the resulting (sparse) score vector (e.g., sorting the positive scoring classes). Efficiency constraints translate to limiting the number of nonzero entries in each row. In the indexing algorithms of this paper, the sum of the entries in each row does not exceed 1. Lemma 1 below states that this restriction does not lose power, among the set of nonnegative matrices, for achieving good rankings.

3.3 Evaluating the Index

We evaluate index learning based on efficiency as well as the quality of classification (accuracy). In large-scale learning, both memory and time efficiency are important, and both at training as well as classification times.[7] Our other goal is to maintain satisfactory accuracy. In our experiments in Section 6.2 (on finite samples), we report on three measures of efficiency: training time T_{tr}, the size of the index learned, denoted by $|\mathbf{W}|$, meaning the number of edges or nonzero weights in the index, and the average number of edges d touched (processed) per feature during classification (a measure of work/speed during classification time). We next describe our classification accuracy measures.

We use the standard accuracy (i.e., one minus zero-one error), here denoted R_1, as well as other measures of ranking quality. R_1 allows us to compare to other published results. A method for ranking classes, given an instance x, outputs a sorted list of zero or more classes. In addition to weighted indices, we describe other methods for ranking the classes in Section 5. An instance may belong to multiple classes in some tasks (two of our data sets in Figure 8). To simplify evaluation and presentation, in this paper we only consider the highest ranked true class. Let k_x be the rank of the highest ranked true class after presenting instance x to the system. Thus $k_x \in \{1,2,3,\cdots\}$. If the true class does not appear in the ranked list, then $k_x = \infty$. We use R_k to denote *recall at (rank) k*, which measures the proportion of (test) instances for which one of the true classes ended in the top k classes:

$$R_k = \text{ recall at } k = E_x[k_x \leq k],$$

where E_x denotes expectation over the instance distribution and $[k_x \leq k] = 1$ iff $k_x \leq k$, and 0 otherwise (Iverson bracket). So we get a reward of 1 if the true class is within top k for a given instance, 0 otherwise, and R_k is the expectation. In our experiments, we will report on (average) recall at rank 1, R_1, and recall at rank 5, R_5, on held-out sets. R_1 is simply the standard accuracy, that is,

6. Feature ratings, can be incorporated in a diagonal matrix $\mathbf{R}$, where $\mathbf{R}[\mathbf{i},\mathbf{i}] = \mathbf{r_i}$ (the rating of feature i) and $\mathbf{R}[\mathbf{i},\mathbf{j}] = \mathbf{0}$, when $i \neq j$. Obtaining the scores would then be $\mathbf{v_x^T RW}$.

7. Note that in online learning, there isn't a sharp separation between the training and testing phases.

the highest ranked class is assigned to the instance, and R_1 measures the proportions of instances to which the true class was assigned.

We also report on the harmonic (mean) rank (HR) (reciprocal of mean reciprocal rank or MRR), defined as:

$$MRR = E_x \frac{1}{k_x}, \text{ and } HR = MRR^{-1}.$$

MRR gives a reward of 1 if a correct class is ranked highest, the reward drops to 1/2 at rank 2, and slowly goes down the higher the k (the lower the rank). If the right class is not retrieved, the reward is 0. MRR is the expectation or the empirical average of such reward over (test) instances, and we simply invert it to get a measure of ranking performance, the harmonic rank HR. The lower the HR, the better, and it has a minimum of 1 (rank 1 is best). MRR is a commonly used measure in information retrieval, such as in question answering tasks (e.g., Radev et al., 2002). In our experiments, we report the HR values so that the reader can quickly get an impression of the average class-ranking performance of the various methods.

Both R_k and MRR are appropriate for settings in which we value better ranks *significantly* more than worse ranks. Thus, if an index is perfect half the time, that is, ranks the correct class of the given instance at top (rank 1) half the time, but fully fails the rest of the time, that is, does not retrieve the correct class at all, then its HR value is 2. However, for an index that always retrieves the correct class, but ranks it third, the HR value is worse, at 3. Note that one could raise the fraction $\frac{1}{k_x}$ to a different exponent (instead of 1) to shift the emphasis in one direction or another. R_k does not reflect the quality of ranking within top k, and it simply cuts the reward off if the right class is outside top k. HR is a smoother measure. Our evaluation measure are from the point of view of an instance to be classified. This is appropriate with large numbers of classes and in many applications, such as personalization or text prediction, in which a given instance (a query, a page, etc.) should be classified into one or a few classes that the system is confident about. In a number of information retrieval tasks such as question answering and document retrieval, the extra emphasis on higher ranks is well motivated. We expect that the situation would be similar for typical many-class problems, such as text categorization. The common precision and recall measures used in machine learning are often computed from the point of view of a class: for each class, the instances are ranked according to the classifier's scores for the class. This is especially appropriate when we are interested in performance on a single class at a time. For instance, when we seek to rank or filter instances based on their degree of membership in a given class of interest (e.g., a news topic). Our indexing techniques are more appropriate for the problem of obtaining good rankings per instance, similar to some other multiclass ranking algorithms (e.g., Crammer and Singer, 2003a). However, existing techniques for improving precision/recall for imbalanced classes may be applicable (e.g., Li et al., 2002). We conclude this section with a simplifying property of non-negative matrices, for the purposes of ranking.

Lemma 1 *Let* $\mathbf{W}$ *be the non-negative matrix corresponding to an index (features correspond to the rows and classes are the columns). The ranking that* $\mathbf{W}$ *produces on nonzero scoring classes is not changed under positive scaling, that is,* $\alpha\mathbf{W}$*, for* $\alpha > 0$*, produces the same ranking.*

Proof The score for each class is obtained in the vector $\mathbf{v_x^T W}$. Therefore, the ranking obtained from $\mathbf{v_x^T} \alpha\mathbf{W} = \alpha \mathbf{v_x^T W}$, is the same as the ranking in the vector $\mathbf{v_x^T W}$, when $\alpha > 0$ and all entries in $\mathbf{v_x^T W}$ are non-negative. ■

The lemma implies that optimal matrices, for the objective of say maximizing R_1 on the training set, among non-negative matrices in which the entries in each row sum to at most 1.0, exist. The indexing algorithms presented in this paper learn non-negative weight matrices.

3.4 Computational Complexity of Index Learning

Can we efficiently compute an index achieving maximum training accuracy given any finite set S of instances? If we constrain the outdegree of each feature to be below a given constant (motivated by space and time efficiency), then the corresponding decision problem is NP-hard under plausible objectives such as optimizing accuracy (R_1):

Theorem 2 *The index learning problem with the objective of either maximizing accuracy (R_1) or minimizing HR on a given set of instances, and with the constraint of a constant upper bound (e.g., 1) on the outdegree of each feature is NP-Hard.*

The proof is by a reduction from the minimum cover problem (see Appendix A). A problem involving only two classes is shown NP-hard. We do not know whether the indexing problem is approximable in polynomial time however, or whether removing the constraint on the outdegree alters the complexity. Linear programming formulations exist with continuous objectives and no explicit outdegree constraint (Madani and Connor, 2007; Madani and Huang, 2008).

The next section describes very efficient online algorithms that perform well in our experiments. We motivate our choices in the algorithm design, but leave theoretical guarantees to future work.

4. Feature Focus Algorithms for Index Learning

Figure 4 present our main index learning technique. After first giving a quick overview of the approach, we motivate the choices in the design of the algorithm in the rest of this section.

On a given instance, after the use of the index for scoring and ranking (an invocation of RankedRetrieval), if a measure of *margin* (to be described shortly) is not large enough, an update to the index is made. The margin is the score obtained by the true class, minus the highest scoring incorrect (negative) class (either of the two scores can be zero). Our index learning algorithms may be best described as performing their updates from the features' side or features' "point of view" (rather than the classes' side or class prototypes), and hence we name the whole family *feature-focus* algorithms. As we will explain, this design was motivated by considerations of efficiency (Sections 4.1 and 4.4). The basic question for each feature is to which subset of classes it should connect (possibly none), and with what weights. Figure 4(d) gives a generic feature updating scheme and Figure 4(c) gives the instantiation we use in our experiments. Initially, all weights are zero. Note that when a weight is zeroed, the connection is removed. This means that, in our index implementation, the list entry corresponding to the edge is removed from the list of the edges of the feature.

We next motivate the design choices in FF. The problem of what each feature in isolation should do during learning turns out to be helpful and we first explore and discuss this single feature case. We then present the IND(ependent) method, a baseline in which effectively on every instance every feature updates. We then motivate mistake-driven updating, and in particular the use of margin.[8]

8. All the examples given to illustrate various aspects make the assumption of Boolean feature values, but the feature-focus algorithm as presented works with the more general nonnegative values.

/* The FF Algorithm */
Algorithm FeatureFocus(x, w_{min}, d_{max}, δ_m)
1. RankedRetrieval(x, d_{max})**. /* retrieve/score */**
2. Compute the margin δ**:**
$\delta = s_{c_x} - s'_x$**, where** $s'_x = \max_{c \neq c_x} s_c$**.**
3. If $\delta > \delta_m$**, return. /* update not necessary */**
4. Otherwise, for each active $f \in x$**:**
/* update active features' connections */
4.1 FSU(x, f, w_{min})**.**
(a)

Algorithm RankedRetrieval(x, d_{max})
/* initially, for each class c**, its score** s_c **is zero */**
1. For each active feature f **(i.e.,** $\mathbf{v_x}[f] > 0$**):**
For the first d_{max} **classes with highest connection weight to** f**:**
1.1. $s_c \leftarrow s_c + (r_f \times w_{f,c} \times \mathbf{v_x}[f])$
2. Return those classes with nonzero score, ranked by score.
(b)

/* Feature Streaming Update (allowing "leaks") */
Algorithm FSU(x, f, w_{min}) **/* Single feature updating */**
1. $w'_{f,c_x} \leftarrow w'_{f,c_x} + \mathbf{v_x}[f]$ **/* increase weight to** c_x**. */**
2. $w'_f \leftarrow w'_f + \mathbf{v_x}[f]$ **/* increase total out-weight */**
3. $\forall c,\ w_{f,c} \leftarrow \frac{w'_{f,c}}{w'_f}$ **/* (re)compute proportions */**
4. If $w_{f,c} < w_{min}$**, then /* drop tiny weights */**
$w_{f,c} \leftarrow 0, w'_{f,c} \leftarrow 0$
(c)

Algorithm GenericWeightUpdate
Each active feature:
1. Strengthens weight to true class
2. Weakens other class connections
3. Drops weak edges (tiny weights)
(d)

Figure 4: (a) Pseudo-code for the Feature-Focus (FF) learning algorithm. The FF algorithm is invoked on every training instance. This corresponds to steps 2 and 3 in Figure 3(a). (b) The RankedRetrieval procedure for scoring and ranking (copied from Figure 3(b)). (c) Feature streaming update, or FSU: The connection weight of f to the true class c_x is strengthened. Others connections are weakened due to the division. All the weights are zero at the beginning of index learning. (e) Generic weight updating: on each training instance, each active feature strengthens its weight to the true class, weakens its other connections, and drops those that are too weak.

We conclude with a comparison of FF to existing online algorithms, in particular the perceptron algorithm and Winnow. The reader may wish to skip some of these sections at this point and go to the experiments (Section 6) on a first reading.

4.1 Updating for a Single Feature

Assume (training) instances arrive in a streaming fashion (from some infinite source), and assume the single label (per instance) setting. Fix one feature and imagine the substream of instances that have that feature active. Let us consider Boolean feature values only ($\mathbf{v_x}[f] \in \{0,1\}$) here for simplicity. Thus, we basically obtain a stream of observed classes, $<c^{(1)}, c^{(2)}, c^{(3)}, \cdots>$, for the given feature. Ignoring other features for now, and considering efficiency constraints, to which classes should this feature connect to, and with what weights? We next argue that our objective of a good ranking, subject to efficiency, reduces to computing the proportion in the sequence for those classes (if any) that exceed a desired proportion threshold.

In this single feature case, classes are ranked by the weight assigned to them by the feature. The constraint (of space efficiency) is that the feature may connect to only a subset of all possible classes, say d_{max} at most. The question is how the feature should connect so that an objective such as R_k or HR (harmonic rank) is maximized. We will focus on the scenario where the stream of classes is generated by an iid drawing from a fixed probability distribution.

It is not hard to verify that the best classes are the d_{max} classes with the highest proportions in the stream, or the highest $P(c)$ if the distribution is fixed and known (more precisely, $P(c|f)$, but f is fixed here) and the ranking should also be by $P(c)$. For a finite sequence on which we are allowed to compute proportions before having to connect the feature, this can easily be established.

Lemma 3 *A finite sequence of classes is given (class observations). To maximize HR, when the feature can connect to at most k different classes, a k highest frequency set of classes should be picked, that is, choose S, such that* $|S| = k$ *and* $S = \{c|n_c \geq n_{c'}, \forall c' \notin S\}$*), where* n_c *denotes the number of times c occurs in the sequence. The classes in S should be ordered by their occurrence counts to maximize HR. The same set maximizes* R_k*.*

Proof This can be established by a simple "swapping" or "exchange" argument. We look at the sum of rewards over the sequence rather than averages, as the sequence length is fixed. Consider maximizing R_k first. Let n_c denote the number of times class c appears in the sequence. For any chosen set S of size k, a pair of classes (c, c') is out of order if $n_c < n_{c'}$, but $c \in S$, and $c' \notin S$. Then R_k for S is improved if c is replaced by c', the improvement is $n_{c'} - n_c$. Similarly HR is improved for an ordered set S if a pair like above exists (improvement of $(n_{c'} - n_c)\frac{1}{j}$, where j denotes the rank of c in S), or a pair within the chosen set is out of order (improvement of $(n_{c'} - n_c)(1/j - 1/j')$, where $j', j' > j$, denotes the old rank of c'.). ■

For unbounded streams generated by iid drawing of classes from a fixed distribution over a finite number of classes, the empirical proportions of classes, over the sequence seen so far (of length at least k), take the place of the counts, in order to maximize expected HR or expected R_k on the unseen portion of the sequence.

We will use FSU (*Feature Streaming Update*, Figure 4(c)) in our main feature-focus algorithm. An FSU update takes at most two list traversals (involving finding or inserting the connection). With d connections per feature, a full update on an instance takes $\tilde{O}(d|x|)$. Note that when features are Boolean, FSU simply computes edge weights that approximate the conditional probabilities $P(c|f)$ (the probability that instance $x \in c$ given that $f \in x$ and FSU is invoked). Since the weights are between 0 and 1 and approximate probabilities, it eases the decision of assessing importance of a connection: weights below w_{min} are dropped at the expense of some potential loss in accuracy. FSU keeps total counts (w'_f and w'_{f,c_x}, which we will describe and motivate later). Note that w_{min} effectively bounds the maximum outdegree during learning to be $\frac{1}{w_{min}}$. We note that this space efficiency of FSU is central to making feature-focus algorithms space and time efficient (see Section 6.3.1). Given that FSU zeros some weights during its computation, it is instructive to look at how well it does in approximating proportions for the (sub)stream of classes that it processes for a single feature. This gives us an idea of how to set the w_{min} parameter and what to expect. Appendix A presents synthetic experiments and a discussion. To summarize, when the true probability (weight) w of interest is several multiples of w_{min}, with sufficient sample size, the chance of dropping it is very low (the probability quickly goes down to 0 with increasing $\frac{w}{w_{min}}$), and moreover, the computed weight is also close to the true conditional. See Section 6.3.3 on the effect of choice of $w_{min} \in \{0.001, 0.01, 0.1\}$ on accuracy on several data sets.

4.1.1 UNINFORMATIVE FEATURES, ADAPTABILITY, AND DRIFTING ISSUES

In FSU, we keep and update two sets of weights, the edge weights $w_{f,c}$ (not greater than 1), $w'_{f,c}$, as well as total weight w'_f. In case of binary features ($\mathbf{v_x}[f] = 1$), we can simply think of w'_f as total count of times FSU has been invoked for the feature, and $w'_{f,c}$ as an under-estimate of the co-occurrence count in that stream ($w'_{f,c}$ can be less than the co-occurrence count, as it is reset to 0 if the edge is dropped). Note that if c_x is not already connected (for example in the beginning), $w_{f,c}$ and $w'_{f,c}$ are 0. An important point is that total weight w'_f is never reduced. This is useful as a way of down-weighing uninformative features (such as "the"). Thus, due to edge dropping, we may have the sum of proportions remain less than 1, $\sum_c w_{f,c} < 1$, even when $w'_f > 0$. We have found this alternative slightly better in our experiments than the case in which $w'_f = \sum_c w'_{f,c}$ (i.e., when w'_f is kept as the exact sum of the weights). See Section 6.3.5.

In case of non-Boolean feature values, similar to perceptron and Winnow updates (Rosenblatt, 1958; Littlestone, 1988), the degree of activity of the feature, $\mathbf{v_x}[f]$, affects how much the connection between the feature and the true class is strengthened. We could use a *learning rate*, a multiplier for $\mathbf{v_x}[f]$, to further control the aggressiveness of the updates. We have not experimented with that option.

Note also that as w_f grows, the feature may become less adaptive, as a new class will have to occur more frequently to obtain a strong weight ratio with respect to w_f. In particular, after $w_f > \frac{1}{w_{min}}$, a new class will be immediately dropped.[9] For long-term online learning, where distributions can drift (nonstationarity), this can slow or stop adaptation, and updates that effectively keep a finite memory or history are more appropriate. Note also that, if the same training instances can be seen multiple times (e.g., in multiple passes on finite data sets), with w_f growing, the fitting capability of the algorithms is curbed. This may be desired as a means of overfitting prevention. Other indexing updates have been developed, offering various trade-offs (see our discussion in Section 4.4, and Madani and Huang 2008, and Madani et al. 2009, in particular for a simple update appropriate for nonstationarity).

Before describing the main feature-focus algorithm, we describe a baseline algorithm we refer to as IND(ependent). This algorithm can be implemented in an offline (batch) manner. It is based on computing the conditionals $P(c|f)$.

4.2 Always Updating (the IND Algorithm)

One method of index construction is to simply assign each edge the class conditional probabilities, $P(c|f)$ (the conditional probability that instance $x \in c$ given that $f \in x$). This can be computed for each feature independent of other features. We refer to this variant as the IND ("INDependent") algorithm (Figure 5). Features are treated as Boolean here ($\mathbf{v_x}[f] \in \{0,1\}$). After processing the training set (computing counts and then conditional probabilities), only weights exceeding a threshold p_{ind} are kept. The use of p_{ind} not only leads to space savings, but also can improve accuracy significantly. The best threshold p_{ind} (for improving accuracy) is often significantly greater than 0 (see Section 6.3.7). In our experiments with IND, we choose the best threshold by observing performance on a random 20% subset of the training set. We thus implemented the IND algorithm

9. At this point, updates can only affect classes already connected, and updates may improve the accuracy of their assigned weights, though there is a small chance that even classes with significant weights may be eventually dropped (this has probability 1 over an infinite sequence!). In any case, at this point or soon after, it is possible to stop updating. In our experiments, with finite data and small number of passes over the data sets, this was not an issue.

Algorithm IND(S, p_{ind}) /* IND algorithm */
1. For each instance x in training sample S:
1.1 For each $f \in x$: /* increment counts for f */
1.1.1 $n_f \leftarrow n_f + 1$
1.1.2 $n_{f,c_x} \leftarrow n_{f,c_x} + 1$
2. Build the index: for each feature f and class c:
2.1 $w \leftarrow \frac{n_{f,c}}{n_f}$**.**
2.2 If $w \geq p_{ind}$**,** $w_{f,c} \leftarrow w$**. (otherwise** $w_{f,c} \leftarrow 0$**.)**

Figure 5: Pseudo-code for the IND(ependent) algorithm, implemented for the case of Boolean features only. The choice of p_{ind} affects accuracy significantly, and is picked using a held out set (see Section 4.2).

as a batch algorithm, that is, we computed the weights $P(c|f)$ exactly, not in an online streaming manner described[10] for FSU. The exact computation can be done on the relatively smaller data sets. IND is in fact the fastest algorithm on the smaller data sets, since the count updates are simple and there is no call to index retrieval during training. This counting phase for index construction can also be distributed. On larger data sets, IND runs into memory problems and becomes very slow during training, due to many features keeping connections to too many classes.[11] This aspect points to the importance of space efficiency for large-scale learning.

The IND algorithm, in its independent computations of weights for each feature, has similarities with the multiclass Naive Bayes algorithm (e.g., Rennie et al. 2003). Major differences include the computation of $P(f|c)$ (the reverse) in plain multiclass Naive Bayes, and that for classification, we are summing the weights (instead of multiplying under the independence assumption), similar to some techniques for expert opinion aggregation (Genest and Zidek, 1986; Cesa-Bianchi et al., 1997). We have found that summing improves accuracy. See Madani and Connor (2007) for a more detailed comparison to multiclass Naive Bayes. In its independent computation of weights, IND is also similar to inverted index construction using, for instance, TFIDF.

IND offers a nice baseline, but we can potentially do significantly better than computing proportions for each feature independently. Often features are inter-dependent. For instance, features can be near duplicates or redundant. In particular, with increasing feature vector sizes, the accuracy of methods that in effect assume feature independence can degrade significantly.

4.3 Mistake-Driven Updating Using a Margin (the FF Algorithm)

FF adds and drops edges and modifies edge weights during learning by processing one instance at a time,[12] and by invoking a feature updating algorithm, such as FSU. Unlike IND, FF addresses feature dependencies by not updating the index on every training instance. Equivalently, a feature updates its connection on only a fraction of the training instances in which it is active. This is motivated and explained next.

10. In case the instance belongs to multiple classes, step 1.1.2 is executed for each true class.
11. However, note that the FSU algorithm can be instead employed here to keep memory consumption in check.
12. The feature and class sets can also grow incrementally.

4.3.1 WHEN TO UPDATE?

FSU should not be invoked on every training instance. In particular, *"lazy"* or mistake-driven updating (not updating all the time) can, to some extent, address issues with feature dependencies. It can, for example, avoid over counting the influence of features that are basically duplicates by learning relatively low connection weights for each such feature (similar to a rational for mistake driven updates in other learning algorithms such as the perceptron). We next give a simple scenario, *case 1*, to demonstrate accuracy improvements that can be obtained by lazy updating.

Case 1. Imagine the simple case of two classes, c_1 and c_2, and two Boolean features, f_1 and f_2. Assume f_1 is perfect for c_1, $P(c_1|f_1) = P(f_1|c_1) = 1$, but that f_2 appears in instances of both classes, and $P(f_2|c_1) = 1$ (i.e., f_2 appears in all instances of c_1), but also $P(f_2|c_2) = 1$. Then, given only f_2, that is, an instance $x = \{f_2\}$ (x contains f_2 only), we want to rank c_2 higher. Now, if say $P(c_1) > P(c_2)$ (c_1 is more frequent than c_2), and we always invoked FSU, f_2 would also give a higher weight to c_1, ranking c_1 higher than c_2 on $x \in c_2$. An optimal solution, for accuracy R_1 or for HR, has the property that f_2 has a higher connection weight to c_2 than to c_1 (with $w_{f_1,c_2} = 0$, an optimal solution satisfies: $w_{f_1,c_1} > w_{f_2,c_2} > w_{f_2,c_1}$.). Now, if FF invoked FSU only when the correct class was not ranked highest, the connection weights in this example would converge to an optimal configuration. To see this, note that as soon as $x \in c_1$ is seen f_1 obtains a weight of 1 to c_1. Next, only updates on $x \in c_2$ will be performed, since c_1 is ranked correctly due to f_1 having a weight of 1 and f_2 keeping some nonzero weight to it. f_2 makes a stronger connection to c_2 than c_1 after at most 2 FSU invocations. R_1 in the optimal case would be 1.0 here, while it can approach 0.5 if we always update. Note that as fewer updates in general mean fewer connections (sparser indices), we may also save in space in this lazy update regime (see Section 6).

On the other hand, if we don't update at all when the right class is at rank 1, we may also suffer from suboptimal performance. This happens even in the case of a single feature. Thus *"proactive"* updating is useful too. The next case elaborates.

Case 2. Consider the single feature case and three classes c_1, c_2, c_3, where $P(c_1) = 0.5$, while $P(c_2) = P(c_3) = 0.25$. Thus c_1 should be ranked highest, for say maximizing R_1. This yields optimal $R_1 = 0.5$, and if we always invoke FSU, this will be the case after a few updates (we will soon get $w_{1,1} \approx 0.5$, and $w_{1,2} \approx w_{1,3} \approx 0.25$). If we don't update when true class is at rank 1, c_2 or c_3 can easily take the place of c_1 when an instance $x \in c_2$ or $x \in c_3$ is presented, but we need to reverse the situation subsequently when $x' \in c_1$ is presented, and instances belonging to c_1 are more frequent. In general, the connection weights from the feature to c_1, c_2, and c_3 will be similar in the purely mistake-driven updating regime, and on sequences that look like the worst case alternating sequence: $c_1, c_2, c_1, c_3, c_1, c_2, \cdots$, the running value of R_1 can approach 0. While random sequences are not as bad, we should still expect significant inferior performance. On randomly generated sample of size 2000 according to above class distribution, averaging over 100 80-20 splits, always (proactive) updating gave an average R_1 performance of 0.479 ± 0.02 (standard deviation of 0.02), while the lazy update gave 0.428 ± 0.09.

Therefore, not updating when the rank of the right class is adequate may cause unnecessary instability in behavior and inferior performance as well. Of course, we desire an algorithm that can perform well in the single feature case. Continued updating even when the true class is ranked highest is akin to keeping a kind of extended memory (in the connection weights).

We strike a balance between the two desirables by using the notion of *margin*. The margin on the current instance is the score of the positive class minus the score of the highest scoring negative

class:

$$\delta = s_{c_x} - s'_x, \text{ where } s_{c_x} \geq 0, s'_x \geq 0, s'_x = \max_{c \neq c_x} s_c.$$

If the margin δ does not exceed a desired margin threshold δ_m, we update[13] (invoke FSU). Note that both s_{c_x} and s'_x can be 0. If we set the margin threshold to 0, we may fit more instances in the training set, and handle situations like case 1, but underperform for case 2 situations. With a sufficiently high margin, updates are always made and case 2 is covered, but fitting power (case 1) can suffer. There is a tradeoff, and a good question is what the best choice of threshold may be? The best choice depends on the problem and the feature vector representation. Individual edge weights are in the $[0,1]$ range, and when the instances are l_2 normalized, we have observed that on average top classes obtain scores in the $[0,1]$ range as well, irrespective of data sets or choice of margin threshold (Madani and Connor, 2007).

Our use of margin is somewhat similar to the use of margin for online algorithms such as perceptron and Winnow (e.g., Carvalho and Cohen, 2006; Crammer and Singer, 2003a), although our particular motivation from considering case 2, "stability" or keeping some "extended memory" for each feature, appears to be different.

4.3.2 RATING THE FEATURES: DOWN-WEIGH INFREQUENT FEATURES

It may be a good idea to down-weigh or eliminate those features' votes that are only seen a few times during learning, as their proportion estimates (connection weights) can be inaccurate and in particular higher than what they should be. Consider the first time FSU is invoked on a feature. After that update, such a feature gives a weight of 1 (the maximum possible) to the class it gets connected to. This is undesired. Of course, how much to down-weigh can depend on the problem, and how feature values are distributed. In our experiments, during scoring of the class, we multiply a feature's score for class c, $w_{f,c}$, by a rating r_f (see Equation 1 in Section 3.2), $r_f = \min(1, \frac{\#_f}{10})$, where $\#_f \geq 1$ denotes the number of times feature f has been seen so far. $\#_f$ is computed only during the first pass over training data. We show that on some problems, this option improves accuracy.

4.4 Summary and Relations to other Methods

The FF algorithm aggregates the votes of each features for ranking and classification. During learning, FF may be viewed as directing a stream of classes to each feature, so that each feature can compute weights for a subset of the classes that it may connect to. The stream, for example with the use of margin, may be hard to characterize and may show drifts during learning: it may initially be those instances in which the feature is active, but later it may correspond to a subsequence of those instances which are somewhat hard to classify. Features may be space constrained: they need to be space efficient in the number of connections they make as well as in computing their connection weights. This efficiency aspect is especially important in large-scale many-class learning.

The FF algorithm has similarities with online algorithms such as Winnow (Littlestone, 1988), as it normalizes (in general weakens some of) the weights, and the perceptron algorithm (Rosenblatt, 1958), as for example the updates are in part additive (ignoring the normalization or weakening). The important difference that changes the nature of the algorithm is that changes to weights are

13. For instances with multiple true classes, the margin is computed for each positive true class. Every active feature is updated for each true class for which its margin is below the margin threshold.

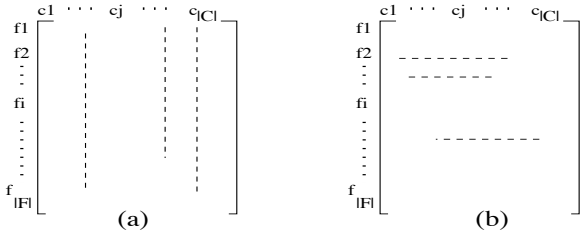


Figure 6: In learning a weight matrix for multiclass learning (here the features corresponding to the rows), prototype methods operate on the columns (part a), for instance in normalizing the (column) weight vectors, while feature-focus methods operate on the rows (part b), for example in ensuring that the number of nonzeros in each row remain within a budget.

done from the side of features, unlike Winnow or perceptron. The Winnow algorithm does the normalization from the side of the classes: each class is represented by a classifier (a class prototype), and each classifier has its feature weights normalized after each update. If normalization is done for all features, many features, whether or not active, get weakened. In a sense, the classifier ranks the features in order of importance for its own concept. A number of learning algorithms in the family of linear classifier learning algorithms, focus on the class side, for example, learning a prototype classifier for each class (e.g., Crammer and Singer, 2003a) (see Figure 6). This is a natural approach for binary classification. In our case, it is the classes whose connections to a feature may be weakened due to one or more classes being strengthened. In the FSU update given in this paper, this weakening happens irrespective of whether a class was ranked high (this aspect is similar to Winnow, but again, for class weights instead of feature weights). Alternative feature-focus updates are possible (e.g., Madani and Huang, 2008). It is best to view each feature as a voter or "expert", and the goal is to obtain good class rankings for each instance by adjusting the votes. A prototype for a class is more appropriate for ranking instances for that class.

To keep memory consumption in check, it seems most direct to constrain features not to connect to more than say 10s of classes, rather than somehow constraining the classes (class prototypes). It appeared harder to us to bound the number of features a class needs, and different classes may require widely varying number of features for adequate performance. See Raghavan et al. (2007) for an exploration of the number of useful features that different (binary) learning problems need for achieving (nearly) maximum accuracy. We also note that in many problems of interest, the number of classes, while large, is significantly smaller than the number of features. In many domains, as the number of classes grows, the number of features tends to grow too (possibly in a proportional manner). In the best of worlds, each feature could be predictive for at most one class. While reality is far from this idealized picture, and we anticipate many interactions, expecting that features may not require high outdegrees for good accuracy, can be a good first assumption. A fruitful future avenue may be exploring this assumption via modeling and developing theoretical arguments.

A second related reason is that constraining the feature outdegrees to remain relatively small (e.g., 10s) appears easier to implement and more time efficient in an online processing regime. Again, a class may require 100s of features and beyond for good performance. Therefore processing the needed classes, to examine importance of features, can take more time, per instance. Finally, we seek rapid categorization per instance, and constraining indegree of classes may not guarantee that

the outdegree of commonly occurring features would be small. Constraining the degrees of classes is not directly related to the average time required for processing an instance. Given that an index is required for efficient classification, that is, efficient access from features to the relevant classes, one would need additional data structures (additional memory) for efficient prototype processing.

For the perceptron update, continued updating can increase weight magnitudes with no bound. This makes designing an effective weight management criterion difficult. False positive classes may obtain negative connections to features they weren't connected to (when ranked higher than the true positive). These extra connections hurt sparsity. Moreover negative connections may not be as useful in the task of ranking multiple classes, to the extent that they are useful in the binary-class case and when learning a single prototype, for ranking instances: in the many-class case, the true class could simply have higher positive weights to the appropriate features. Of course, our discussion does not preclude efficient algorithms that, nevertheless, perform their operations from the class side.

5. Techniques Based on Binary Classifier Training

We compare against hierarchical or top-down training and classification, a commonly used method when a taxonomy of classes, a tree from general classes at the top to specific classes, is available (Dumais and Chen, 2000; Liu et al., 2005). The hierarchical method reduces to one-versus-rest classification when the classes are flat (when there is just one level), which is another common method for multiclass classification (e.g., Rifkin and Klautau, 2004). We compare against one-versus-rest on relatively small sets, to see how indexing performs in more traditional classification settings. Note that the FF algorithm, while motivated by many-class learning, is a linear method applicable to few classes and in particular binary classification as well.

The one-versus-rest method simply trains a binary classifier for each class using all the data. During classification, all the classifiers are applied and their scores rather than classification outcomes are used for ranking.[14] We observed no advantage in obtaining probabilities here compared to using raw scores. The one-versus-rest method becomes quickly inefficient, at both training and classification times, as the number of classes increases (as all the classifiers need to be applied to a given instance).

Linear classifiers such as support vector machines (SVMs) often perform the best in very high dimensional problems such as text classification (Lewis et al., 2004; Sebastiani, 2002). We tested perceptrons and SVMs in one-versus-rest and top-down methods. We use single pass and multiple pass perceptrons (Rosenblatt, 1958) as well as committees of them. Here, each perceptron in the committee is represented as a sparse vector and random weight initialization, in $[-1, 1]$, is used when a new feature is added to the prototype. Unless specified, we run the perceptron learning algorithm until the 0/1 error on training is not improved (computed at the end of each pass), for 5 consecutive passes. Perceptron committees often obtain performance close to SVMs (e.g., Carvalho and Cohen, 2006), although their training time can be less.

14. Note that the use of index for classification is one-versus-rest (or "flat" classification), but the index was not obtained by training binary classifiers.

Algorithm TopDownProbabilities(x, c, p, $\tilde{C}_x$)
1. For each class c_i **that is a child of** c**:**
1.1 $p_{c_i} \leftarrow p \times P_{c_i}(x)$**. /* obtain probability */**
1.2 If $p_{c_i} \geq p_{min}$
1.2.1 $\tilde{C}_x \leftarrow \tilde{C}_x \cup \{(c_i, p_{c_i})\}$
1.2.2 TopDownProbabilities(x, c_i, p_{c_i}, $\tilde{C}_x$)

Figure 7: Pseudo-code for top-down classification. $P_{c_i}(x)$ denotes the probability assigned to x by the classifier trained for c_i in the tree. For each instance x, the first call is TopDownProbabilities(x, root, 1.0, {}).

5.1 Hierarchical Training and Classification

Briefly, hierarchical training works by first training classifiers for the first level classes in a one-vs-rest manner (e.g., Dumais and Chen, 2000). Then the same procedure can be repeated for the children of each class residing in the 2nd level (in general, the level below), training each classifier only on the instances that belong to one of the siblings. Only the classifiers for the top level classes will be trained on all the instances. For ranking and categorization using the hierarchical approach, we use classifier probabilities. We obtain probabilities from classifier scores by the method of sigmoid fitting (Platt, 1999). This may require additional training time for improved accuracy. In the experiments, we report on the effect of increasing the number of sigmoid-fitting trials on one of the data sets (Reuters RCV1).

During classification, whenever a classifier is applied, we use the probability it assigns. The probabilities are multiplied along a path top-down (Figure 7). A path of candidate classes is terminated if the probability falls under some threshold p_{min}. All the classifier at the first level (corresponding to the classes at the top level) are applied to a given instance. During test time, we tried several thresholds: $p_{min} = 0.05 + 0.05k, k = 1, 2, \cdots$, and report results on the threshold giving highest accuracy R_1. All our ranking methods are evaluated on the deepest classes an instance is assigned to. For the evaluation of the top-down method, from the list of candidates obtained for a given test instance, any class whose child is also in the list is removed, and the remaining classes are sorted by their assigned probabilities. For the list of the true classes of the test instance, again only those true classes with no child in the list are kept. Then R_1, R_5 and HR are computed (for the highest ranked true positive class).

We note that if we don't use the probabilities and ranking, that is, use class assignments to follow a path, the classification performance greatly suffers. This is since classifiers (when having to assign a class) in the higher levels can make "premature" false positive and false negative mistakes (and false negative mistakes are very costly). This inferior performance has been noted before too (e.g., Dekel et al., 2003).

6. Experiments

In this section, after describing the data sets we use and the experimental set up, we report on comparisons with other approaches. We then report on several ablation experiments as well as comparisons to the simpler IND algorithm and a previous (unweighted) indexing method. We conclude the section with experiments on some properties of our index learning method and the indices

Data Sets	$\lvert S\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{C}\rvert$	$E_x\lvert\mathbf{v_x}\rvert$	$E_x\lvert C_x\rvert$
Reuters-21578	$9.4k$	$33k$	10	80.9	1
20 Newsgroups	$20k$	$60k$	20	80	1
Industry	$9.6k$	$69k$	104	120	1
Reuters RCV1	$23k$	$47k$	414	76	2.08
Ads	369k	301k	12.6k	27.2	1.4
Web	70k	685k	14k	210	1
Jane Austen	749k	299k	17.4k	15.1	1

Figure 8: Data sets: $|S|$ is number of instances, $|\mathcal{F}|$ is the number of features, $|\mathcal{C}|$ is the total number of classes, $E_x|\mathbf{v_x}|$ is the average (expected) number of unique active features per instance (avg. vector size), and $E_x|C_x|$ is the average number of class labels per instance.

learned (average class indegrees, performance on the training data, ...), and we give a few examples of the learned connections.

Figure 8 presents the data sets that we use, shown in order of class size $|\mathcal{C}|$. The first 6 are text categorization, and the last is a word prediction task. Ads refers to a text classification problem provided by Yahoo! Web refers to a web page classification into Yahoo! directory of topics. Jane Austen is 6 online novels of Jane Austen, concatenated (obtained from project Gutenberg (`http://www.gutenberg.org/`). The others are commonly used text categorization data sets (Lang, 1995; Rose et al., 2002.; Lewis et al., 2004).

On the first three small sets, we compare against one-versus-rest, and our purpose is mainly to compare accuracy. On the next 3, Reuters RCV1, Ads, and Web, we compare against the top-down method. In both the one-versus-rest and top-down methods, we deploy either single perceptron training, committee of 10 perceptrons, or a fast algorithm for learning linear SVMs (Keerthi and DeCoste, 2005). We could not run the SVM on the Web data as it took longer than a day, and had to limit our SVM experiments on Ads.[15] For the final word prediction data, the task is to predict each word given features derived from the surrounding words in the sentence. For this problem, since the classes (the words) do not form a hierarchy and one-versus-rest is too inefficient, we only show performance of the indexing method.

All instance vectors are l_2 (cosine) normalized. For text categorization data, the features are standard unigram or bigram words and phrases. The feature vectors were obtained from publicly available sources in the cases of Reuters RCV1 (Lewis et al., 2004), and the newsgroups (from Rennie[16]). For RCV1, we used the training split only (23k documents) to be able to experiment with the slower algorithms. We obtained the Ads and Web data sets from Yahoo! For the web data set, to obtain a sufficient number of instances per class, we cut the taxonomy at depth 4, that is, we only considered the true classes up to depth 4. To simplify evaluation, we used the lowest true class(es) in the hierarchy the instance was classified under at test time. Thus an instance in Reuters RCV1 corpus on average is assigned two true classes. We note that in many practical text

15. There has been further advances on speeding up linear SVM training algorithms since the submission of this paper (e.g., Shalev-Schwartz et al., 2007; Hsieh et al., 2008). The perceptron training timings in our tables may be a better indicator of the training times for the more recent algorithms.

16. Obtained from people.csail.mit.edu/jrennie/20Newsgroups.

categorization applications such as personalization, classes at the top level are too generic to be informative/useful. For top-down training, we trained the classifier on the internal classes as well. Web and Ads had just over 20 classes in the first level (after root), while Reuters RCV1 has two (we used both the Industry and Topic trees). The Jane Austen (word prediction) data set was obtained by processing Jane Austen's six online novels: the surrounding neighborhood of each word, 3 words on one side, 3 on the other, and their conjunctions, provided the features (about 15 many).

We report on the average performance over ten trials. In each trial a random 10% of the data is held out. The exception is the newsgroup data set where we use the 10 train-test splits by Rennie et al. (2003), each 80-20, and we used their vector representations, to be able to compare directly with their results. We used a 2.4GHz AMD Opteron processor with 64 GB of RAM, with light load.

Figures 9 and 10 present the algorithms' performance under both accuracy and efficiency criteria. As a simple baseline, we report the performance of FrequencyBaseline (FB) as well, which ranks classes simply based on the frequency of the classes in the training data set.

For the FF algorithm, we used w_{min} =0.01 for the minimum weight threshold during learning, and d_{max} =25 (max-outdegree during look up). Note that d_{max} of 25 means a class is retrieved as along as it is within the first 25 highest weight connections of some active feature, even if its weight is not much higher than w_{min}. During training, the FF algorithm looks for a true (positive) class within the first 50 top scoring classes. If it is not found, the score of the positive class is assumed 0. We report on performance after pass 1 with 0 margin threshold ($\delta_m = 0$), as well as best performance in R_1 within the first 10 passes, with $\delta_m \in \{0, 0.1, 0.5\}$. We did not optimize on the choice of these parameters, for example, we may do better for lower d_{max} values (see Section 6.3.2). Note that a δ_m value above 0.5 basically means to update on most instances as index edge weights are less than 1, and thus class score differences tend not to be much higher than 1.0, when instances are l_2 normalized. For the SVM, we report the best performance in R_1 over the regularization parameter $C \in \{1, 5, 10, 100\}$ for the first three small data sets and $C \in \{1, 10\}$ for the large ones. Often $C = 1$ and 10 suffices (accuracies are close). There are 10 perceptrons in the committee (often, 5 to 20 suffices for attaining much of the accuracy).

6.1 Accuracy Comparisons

We first observe that the FF algorithm is competitive with the best of others. In particular it achieves the highest R_5 in 5 of the 6 categorization domains, and the highest average R_1 in 4 of 6 cases. We have observed that comparison based on the HR results often yields similar rankings of the algorithms tested as does R_1. We limit the discussions to R_1 and R_5. In particular R_1 (plain accuracy or one minus zero-one error) is a simple commonly used performance measure. For the classifier based methods, observe that there is a good separation from perceptron to SVMs, suggesting that the classification tasks are challenging. The performance of FF on newsgroup ties the best performance achieved by Rennie et al. (2003), and they used special feature vector representations, for the linear SVM as well as their methods, to reach that performance.[17] In the case of RCVI, for top-down training, we experimented with using a fixed sigmoid, that is, no sigmoid fitting, as well as sigmoid fitting using one or more trials of obtaining scores (score-class pairs). When not fitting, we used fixed values of 0 bias and -2 slope: $\frac{1}{1+e^{-2s}}$, where s denotes the score of classifier on the instance.

17. On the industry data set, we found that the classes have similar proportions, close to 0.01. As we keep only 10% for test, and there are only just under 10k instances in the whole set, we see that the performance of Frequency Baseline is very low. The classes with the highest proportion in training are not the classes with the highest proportion on the test set.

	Rank (HR)	R_1	R_5	T_{tr}	d	$\|\mathbf{W}\|$
Small Reuters, 10 classes						
δ_m=0, p=1	1.082	0.860	0.998	0s	4.9	55k
δ_m=0.5, p=1	1.066	0.884 ±0.009	0.997	0s	5	73k
perceptron	1.08	0.871	0.995	4s	10	74k
committee	1.06	0.891	0.999	40s	10	74+
SVM C=1	1.052	**0.906** ±0.009	0.998	11s	10	74+
FreqBaseline	1.6	0.42	0.86	-	-	-
News Groups, 20 classes						
δ_m=0, p=1	1.137	0.798	0.978	2s	10	113k
δ_m=0.5, p=1	1.085	**0.865** ±0.005	**0.987**	2s	10	171k
perceptron	1.229	0.728	0.928	20s	20	189k
committee	1.122	0.830	0.970	220s	20	189+
SVM C=1	1.1020	0.852 ±0.005	0.975	92s	20	189+
FreqBaseline	3.33	0.05	0.25	-	-	-
Industry, 104 classes						
δ_m=0, p=1	1.114	0.861	0.942	4s	16.7	124k
δ_m=0.5, p=3	1.094	**0.886**±0.008	**0.949**	16s	15.8	196k
perceptron	1.488	0.595	0.773	55s	104	330k
committee	1.17	0.816	0.904	610s	104	330+
SVM C=10	1.112	0.872 ±0.009	0.933	235s	104	330+
FreqBaseline	31	0.005	0.03	-	-	-

Figure 9: Comparisons on the smaller data sets. T_{tr} is training time (s=seconds, m=minutes, h=hours), d is the number of "connections" touched on average per feature of a test instance, and $|\mathbf{W}|$ denotes the number of (nonzero) weights in the system (see Section 6.2). The first two rows for each set report on FF, the first row being FF with 0 margin threshold, after one pass (p=4 means trained for four passes). Some example standard deviations for R_1 are also shown.

Thus, at score of 0, the (probability) value obtained is 0.5, and score of 1, the probability is $\frac{1}{1+e^{-1}} \approx 0.73$. When sigmoid fitting, we used one or more 80-20 splits of the training data, trained on 80, obtained the scores on the remaining 20, pooled the scores from different trials and fitted a sigmoid on the points. We then trained the classifier on the whole set. With more trials, we got better results on RCV1, but with diminishing returns, and this takes more time. For Ads, we could run the SVM with no fitting. Committee and perceptron used 3 trials. We performed the binomial sign test to compare the performance of FF (the second row for each data set) against the best of others (this is the SVM result, when available) as follows. We paired the R_k values on the same splits of data, 10 many for each data set, and counted the number of wins and losses of FF. The bold-faced R_1 and R_5 values indicate significance with confidence level $p \leq 0.05$ (i.e., either 9 or 10 wins). We observe that FF is superior with statistical significance in many cases, and only in one case, R_1 on the smallest data set, does it have statistically significant inferior performance.

The competitive and even superior accuracy of the FF algorithm provides evidence that improving class ranking on each instance, in the context of other classes that may be relevant (other retrieved classes), and at the same time, keeping the index sparse, is a good strategy or learning bias for our high performance categorization task. Methods based on binary classifiers can be at

	Rank (HR)	R_1	R_5	T_{tr}	d	$\|\mathbf{W}\|$
Reuters RCV1, 414 classes						
$\delta_m = 0$, p=1	1.181	0.763	0.955	6s	15.1	181k
$\delta_m = 0.1$, p=4	1.164	0.787±0.008	**0.952**	24s	12.9	223k
perceptron	1.418	0.621	0.815	70s	38	760k
committee	1.197	0.769	0.918	750s	36	760k+
C=1,0fit	1.26	0.72	0.89	94s	36	4meg
C=1,1 trial	1.18	0.779	0.936	200s	36	4meg
C=1,3 trials	1.17	0.782	0.937	400s	36	4meg
C=1,4 trials	1.17	0.783 ±0.01	0.939	520s	36	4meg
FreqBaseline	4.58	0.082	0.348	-	-	-
Ads, 12.6k classes						
$\delta_m = 0$, p=1	1.269	0.706	0.892	27s	7.8	814k
$\delta_m = 0.1$, p=4	1.254	**0.725**±0.003	**0.890**	92s	6.7	1meg
perceptron	1.738	0.517	0.642	0.5h+	80	5meg
committee	1.424	0.652	0.758	5h+	80	5meg+
SVM C=10, 0 fit	1.424	0.665 ±0.003	0.774	12h+	80	5meg+
FreqBaseline	35.56	0.012	0.033	-	-	-
Web, 14k classes						
$\delta_m = 0$, p=1	2.22	0.346	0.575	64s	8	1.6meg
$\delta_m = 0$, p=2	2.21	**0.352**±0.007	**0.576**	128s	8	1.5meg
perceptron	6.69	0.098	0.224	1h+	250	14meg
committee	3.78	0.207	0.335	12h+	190	14meg+
FreqBaseline	10.4	0.053	0.126	-	-	-
Jane Austen, 17.4k classes						
$\delta_m = 0$, p=1	2.71	0.272 ±0.002	0.480	40s	8.7	1.5meg
$\delta_m = 0.1$, p=4	2.73	0.279 ±0.002	0.462	160s	9.1	1.6meg
$\delta_m = 0.5$, p=4	3.01	0.243 ±0.002	0.425	160s	9.1	1.6meg
FreqBaseline	10.3	0.037	0.15	-	-	-

Figure 10: Comparisons on the larger data sets. T_{tr} is training time (s=seconds, m=minutes, h=hours), d is the number of "connections" touched on average per feature of a test instance, and $|\mathbf{W}|$ denotes the number of (nonzero) weights in the system (see Section 6.2). The first two rows for each set report on FF, the first row being FF with 0 margin threshold, after one pass (p=4 means trained for four passes). Some example standard deviations for R_1 are also shown.

a disadvantage because the task of choosing whether a single class should be assigned or not, in isolation, can be error-prone, especially with large numbers of classes. Classifier scores can be used for ranking classes, but the classifiers were not obtained with the objective of a good ranking of the classes for each instance (and their scores may not be calibrated):[18] a typical binary classifier such as an SVM is trained to yield a separation among instances (a good class prototype). The scores of such a classifier are more suitable for ranking the instances for the corresponding class than ranking classes for each instance. Top-down classification can help accuracy in that the top classifiers may effectively discover features useful for making general distinctions, and lower-level classifiers can similarly use features for making fine distinctions among a smaller set of sibling classes. On the

18. However, for the one-versus-rest experiments, we also evaluated rankings using the probabilities obtained via sigmoid fitting, instead of using the raw classifier scores, but saw no change or inferior accuracies (not shown).

other hand, top-down classification inherits the problems of one-versus-rest (for the children of every parent node, the problem is akin to one-versus-rest). Furthermore, the errors of the intermediate classifiers along a classification path can add up. Indexing achieves a kind of direct "flat" classification (akin to one-versus-rest, but not via binary classifiers). Features that are directly predictive of classes can be discovered, skipping error-prone intermediary classifiers. On the other hand, distinguishing thousands of classes via a single linear classifier (the index) can be error-prone.[19] It is ultimately an empirical question which of these fairly different learning techniques may outperform others.

6.2 Efficiency and Ease of Use

We see that the training times of the FF algorithm is dramatically lower than others, and the ratio grows with data set size, reaching or exceeding two orders of magnitude.

Our measure of work, d, is the expected number of "connections" touched per feature of a randomly drawn instance during categorization. For example, for the ads data set, on average just under 8 connections (classes) are touched during index look up per feature, or 8×27 total are touched per instance (the average number of features of a vector is 27, see Figure 8), while for top-down ranking, 80 classifiers are applied on average (over 22 at the top level) during the course of top-down ranking/classification. We are assuming the classifiers have a memory-time efficient hashed representation. Again, we see that the indexing approach can have a significant advantage here.

In the case of the index, the space consumption $|\mathbf{W}|$ is simply the number of edges (positive weights) in the bipartite graph. In the case of classifiers, we assumed a sparse representation (only nonzero weights are explicitly represented), and in most cases used a perceptron classifier, trained in a mistake driven fashion as a lower estimate for other classifiers.[20] On the smaller data sets, the difference is not important. However, we see that on the large categorization data sets the classifier based methods can consume significantly more space. We also note that for the FF algorithm, with higher δ_m, the index size increases. This is caused by more updates being performed with higher δ_m, and more updates tends to increase edge additions. This does not appear to increase the work d though.

The FF algorithm is very flexible. We could run it on our workstations for all the data sets (with 2 to 4 GB RAM), each pass taking no more than a few minutes at most. This was not possible for the classifier based methods on the large data sets (inadequate memory). In general, the top-down method required significant engineering effort (encoding the taxonomy structure, writing the classifiers to file for largest data sets, etc). Liu et al. (2005) also report on the considerable engineering effort required and the need for distributing the computation.

6.3 Effects of Various Options and Parameters

In this section, we investigate the effects of various parameters and options on accuracy and efficiency. For each option, we show performance on a subset of data sets to show the difference that using that option can make. In each case, unless otherwise specified, the remaining parameters (such as choice of margin) are as in Figures 9 and 10 for best performance, and as before we report

19. Huang et al. (2008) explore a multi-stage data-driven classification approach, using fast indexing for the first stage.
20. We have observed that the committee of perceptrons can be converted into a single linear classifier by weight averaging after training without degrading accuracy.

	No Constraints	Default Constraints	T_{tr} (single pass)
Small Reuters	0.884 $\pm$0.008	0.884 $\pm$0.008	0s vs 0s
News Group	0.866 $\pm$0.006	0.865 $\pm$0.005	3s vs 2s
Industry	0.889 $\pm$0.009	0.886 $\pm$0.008	9s vs 4s
RCV1	0.787 $\pm$0.007	0.787 $\pm$0.008	$[40s-50s]$ vs 6s
Ads	0.716	0.711	45m vs 27s
Web	0.327	0.347	2h vs 64s
Jane Austen	0.276	0.274	1h vs 41s

Figure 11: No constraints on d_{max} (maximum outdegree) nor w_{min} (w_{min} set to 0), compared to the default settings. Accuracies (R_1 values) are not affected much, but efficiency suffers greatly. The rough training times for a single pass are compared.

averages and standard deviations for 10 random trials of 90-10 splits (except for news groups, for which the 80-20 split is given).

6.3.1 REMOVING EFFICIENCY CONSTRAINTS

We designed the FF algorithm with efficiency in mind. It is instructive to see how the algorithm performs when we remove the efficiency constraints (w_{min} and d_{max}). Note however that such constraints may actually help accuracy somewhat by removing unreliable weights and preventing overfitting.

In these experiments, we set w_{min} to 0 and d_{max} to a large number (1000). We show the best R_1 result for choice of margin threshold $\delta_m \in \{0, 0.1, 0.5\}$, over the first 5 passes, and compare to default values for the efficiency constraints. We observe that the accuracies are not affected. However FF now takes much space and time to learn, and classification time is hurt too. On the web data, for instance, the number of edges in the index grows to 6.5meg after first pass (it was about 1.5meg before). The average number of edges touched per feature grows to 1633, versus 8 for the default, thus 200 times larger, which explains the slow-down in training time.

For the ads, web, and Jane Austen, due to the very long running times, we ran FF for only a few trials, sufficient to convince ourselves that the accuracy does not change (see also next section). We report the result (with or without constraints) from the first pass of a single trial, on the same split of data.

6.3.2 OUTDEGREE CONSTRAINT

Figure 12 shows accuracy against the degree constraint, d_{max}, on the 3 large categorization data sets. We see that accuracy may in fact improve with lower degrees (RCV1 and Web). At outdegree constraint of 3 for RCV1, the number of edges in the learned index is around 80k instead of 180k (for the default $d_{max} = 25$), and the number of classes (connections) touched per feature is under 3, instead of 15 (Figure 10).

In general, it may be a better policy to use a weight threshold, greater than w_{min}, instead of a max outdegree constraint, for more efficient retrieval, as well as more reduction in index size, without loss in ranking accuracy.

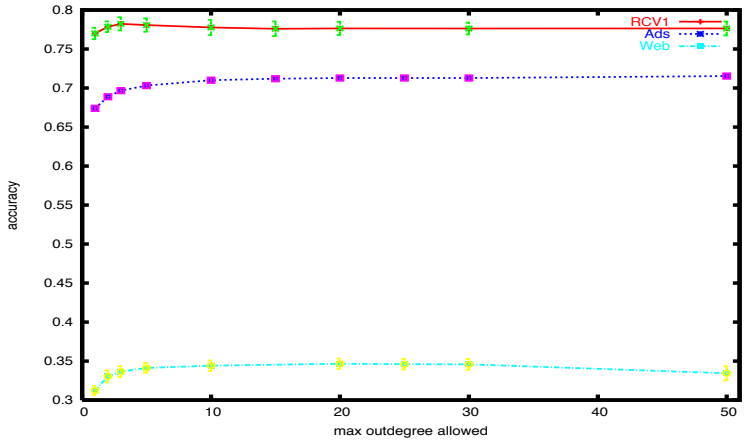


Figure 12: Accuracy (R_1) after one pass against the outdegree constraint.

	0.001	0.01	0.1
RCV1	0.786 $\pm$0.009	0.787 $\pm$0.008	0.761 $\pm$0.008
Ads	0.728 $\pm$0.002	0.725 $\pm$0.003	0.701 $\pm$0.003
Web	0.332 $\pm$0.005	0.352 $\pm$0.003	0.30 $\pm$0.006

Figure 13: The effect of w_{min} on accuracy. We took the best R_1 within the first 5 passes. The standard deviations are also shown. The value $w_{min} = 0.1$ is significantly inferior, while setting w_{min} to 0.001 does not lead to significant improvements.

6.3.3 THE MINIMUM WEIGHT CONSTRAINT

We noted in Section 4.1 that a w_{min} value of 0.01 can be adequate if we expect most useful edge weights to be in say $[0.05, 1]$ range, while a w_{min} value of 0.1 is probably inadequate for best performance. Figure 13 shows the R_1 values for $w_{min} \in \{0.001, 0.01, 0.1\}$ on the three bigger text categorization data sets. Other options were set as in Figure 10, and the best R_1 value within first 5 passes is reported.

Note that while w_{min} =0.1 is inferior, the bulk of accuracy is achieved by weights above 0.1, and $w_{min} \leq 0.01$ does not make a difference on these data sets.

6.3.4 MULTIPLE PASSES AND CHOICE OF MARGIN

Figure 14 shows accuracy (with standard deviations over 10 runs for two plots) as a function of the number of passes and different margin values, in the case of Reuters RCV1. As can be seen, different margin values can result in different accuracies. In some data sets, accuracy degrades somewhat right after pass 1, exhibiting possible overfitting as training performance increases.

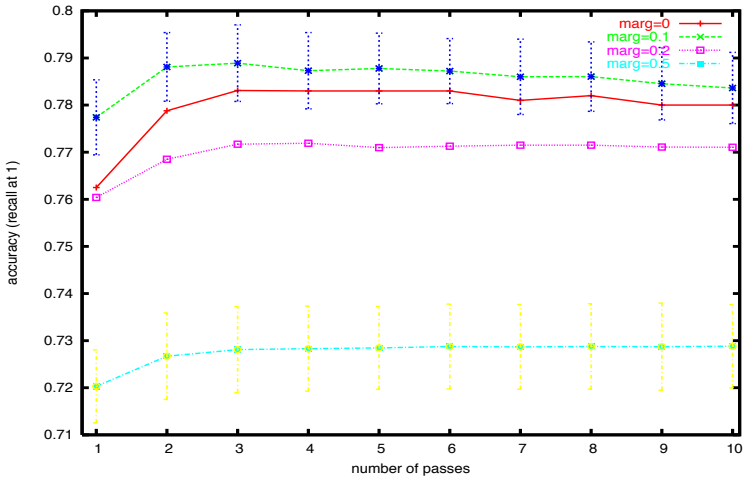


Figure 14: Reuters RCV1: Accuracy (R_1) for margin threshold $\delta_m \in \{0, 0.1, 0.2, 0.5\}$ against the number of passes.

	No Leakage	Allow Leakage
News Group	0.866 $\pm$0.005	0.865 $\pm$0.005
RCV1	0.780 $\pm$0.008	0.787 $\pm$0.008
Ads	0.696 $\pm$ 0.002	0.725 $\pm$ 0.003

Figure 15: On some data sets, allowing weight leakage when dropping edges can significantly improve accuracy.

6.3.5 DISALLOWING WEIGHT "LEAKS"

An uninformative feature such as "the" should give low votes to all classes. However, since the outdegree is constrained for memory reasons, if we imposed a constraint that the connection weights of a feature should sum to 1, then "the" may give significant but inaccurate weights to the classes that it happens to get connected with. Allowing for weight leaks is one way of addressing this issue. Figure 15 compares results. For the NO case in the figure (not allowing), whenever an edge from f to c is dropped, its weight, $w'_{f,c}$, is subtracted from w'_f. Thus $w'_f = \sum w'_{f,c}$ when we don't allow leaks, and $w'_f \geq \sum w'_{f,c}$ when we allow them.

6.3.6 DOWN WEIGHING LITTLE SEEN FEATURES

Figure 16 shows the effect of down weighing infrequent features (the default option, see Section 4.3.2), against treating all features as equal (not using the option). Down-weighing infrequent features can significantly help.

	No Down-Weigh	With Down-Weigh
Newsgroup	0.860 ±0.005	0.865 ±0.005
RCV1	0.758 ±0.007	0.787 ±0.008
Web	0.327 ±0.006	0.352 ± 0.007

Figure 16: Down-weighing infrequent features can significantly improve accuracy.

	IND	Bool FF p=1	best Bool FF	best FF
News Group	0.846 ±0.006	0.860 ±0.006	0.860 ±0.005	0.865 ±0.005
Industry	0.799 ±0.01	0.839 ±0.011	0.867 ±0.008	0.886 ±0.009
RCV1	0.686 ±0.009	0.76 ±0.01	0.780 ±0.008	0.789 ±0.008

Figure 17: Comparisons with IND and the effect of using feature values or treating them as Boolean and no l_2 normalization. The last column (best FF) contains results when default FF (with the use of feature values, l_2 normalization) is used (from Figure 9). Boolean FF with the right margin can significantly beat IND in accuracy, and use of feature values in FF appears to help over Boolean representation.

6.3.7 IND AND BOOLEAN FEATURES

IND treats features independently and as Boolean, but computes the conditionals exactly. Thus IND is similar to Boolean FF with high margin and $w_{min} = 0$, but IND also has a post-improvement step of adjusting p_{ind} (using the training set), which we have observed can improve the test accuracy of IND significantly (in addition to reducing index size). In these experiments p_{ind} was chosen from,

$$\{0.01, 0.02, \cdots, 0.09, 0.1, 0.15, 0.2, 0.25, \cdots, 0.6\}.$$

Here we compare IND against FF with Boolean values (and feature vectors are not l_2 normalized). This allows us to see how much using features values helps, as well as a comparison to a simpler heuristic of computing the conditional probabilities exactly and dropping the small values afterward. Figure 17 shows the results. For Newsgroup, Industry and RCV1, the best value of p_{ind} was respectively $0.01, 0.1$, and 0.3. To see the effect of edge removal on the accuracy of IND, if we chose $p_{ind} = 0$ (did no edge removal), we would get R_1 averaging below 0.58 on RCV1 (instead of current 0.69).

To achieve the best performance with Boolean features for FF on newsgroup, we had to raise the margin threshold to 7.0. Margin threshold of 1 or below gave significantly inferior results of 0.82 or below. Note that the scores that the classes receive during retrieval can increase significantly with Boolean features (compared to using the feature values in l_2 normalized vector representation).

We conclude that IND can be significantly outperformed by FF with an appropriate margin.

6.4 Other Experiments

Here, we first compare to an older indexing method (Madani et al., 2007) and then report and discuss some properties of the FF learning algorithm, such as the training performance, average scores of the top class, and a few example connections learned.

	No Classifiers	With Classifiers	FF
News Group	0.681 ±0.007	0.768 ±0.006	0.86
Industry	0.658 ±0.009	0.795 ± 0.01	0.88± 0.008

Figure 18: The performance of the non-ranking indexer algorithm, learning an unweighted index, with and without classifiers (first two columns) (Madani et al., 2007). The goal of improving class rankings, via learning a weighted index, simplifies indexing and improves classification accuracy.

6.4.1 COMPARISON TO OLDER INDEX LEARNING

The first idea for use of an index was to drastically lower the number of candidate classes to a manageable set when classifying a given instance, say 10s, and then use classifiers, possibly trained using the index as well (for efficient training), to precisely categorize the instances (Madani et al., 2007). Here, we briefly compare using that method, which we will refer to as *unweighted* indexing, against our current FF method. We have already noted that (binary) classifiers appear inferior for class ranking, especially as we increase the number of classes, in our comparisons in one-versus-rest experiments. Here, we present results showing that adding an intermediate index trained as described by Madani et al. (2007) does not improve accuracy. Furthermore, FF is significantly faster and easier to use.

The unweighted indexing algorithm of Madani et al. (2007) uses a threshold t_{tol} during training and updates the index only when more than t_{tol} many false positive classes are retrieved on a training instance or when a false negative occurs. In that work, class-feature weights are computed only to decide whether a connection or an edge should go into the index. We report accuracy under two regimes when we test unweighted indexing: (1) as a baseline, when only using the class-feature weights (without training classifiers), (2) when classifiers are also trained, here committee of perceptrons, trained in an online manner in tandem with the learning of the index, and the classes are ranked using the scores of the retrieved classifiers on the instance. For further details on that algorithm, please refer to Madani et al. (2007). Note that if we use the classifiers for direct classification (and not ranking) we obtain significantly inferior accuracy.

Figure 18 shows the results on the newsgroup and Industry data sets. When using no classifiers, we obtained the best R_1 performance with $t_{tol} = 5$ (out of $t_{tol} \in \{2, 5, 20\}$) on the newsgroup and Industry data sets. The accuracy improves with more passes, but reaches a ceiling in under 20 passes, and we have reported the best performance over the passes. With the addition of classifiers, the best R_1 is obtained when we don't use the indexer (see Figure 9), but the results from using the indexer can be close as the number of classes grows and with tolerance set in 10s. We have shown the result for t_{tol} =5 for newsgroup, and t_{tol} =20 for Industry. We observe that we require classifiers for the unweighted index learning method, to significantly improve accuracy, and the combination still lags behind FF in accuracy.

We note that while unweighted indexing without classifier training is fast, classifier training adds significant space and time overhead. Training was an order of magnitude slower than FF on the two data sets we reported on, and the classifiers also require 10 or more training passes to reach best performance.

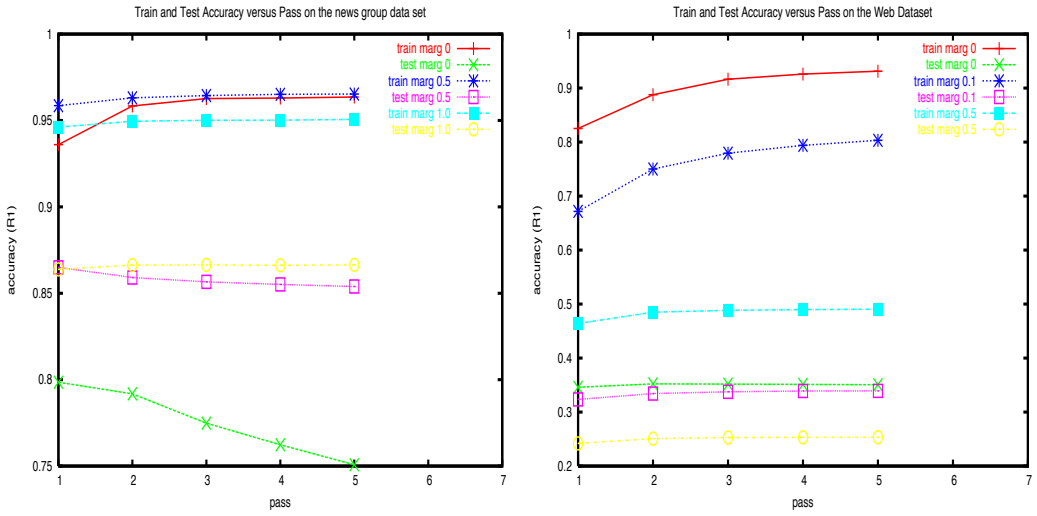


Figure 19: Train and test accuracy versus the number of passes, on the newsgroup (left) and web (right) data sets. Increasing the margin threshold can help control overfitting, but may not result in best test performance.

6.4.2 TRAINING VERSUS TEST PERFORMANCE OF FF

Figure 19 shows the train and test R_1 values as a function of pass. For the training performance, at end of each pass, the R_1 performance is computed on the same training instances rather than on the held-put sets. The higher the margin threshold, the less the capacity for fitting and therefor the less the possibility of overfitting. In the case of the newsgroup, we see that we reach the best performance with a relatively high margin threshold of $\delta_m \approx 1$, and the test and training performance remain roughly steady with more passes, unlike the case for $\delta_m = 0$. For the web data set, we see that the difference between train and test performance also decreases as we increase the margin threshold, but the best test performance is obtained with margin threshold of 0.

6.4.3 LEARNING CLASS PROTOTYPES

FF does not necessarily learn good (binary) classifiers or class prototypes, that is, the incoming weights into a class c_i (the vector $(w_{1,i}, \cdots, w_{f_{|F|},i})$), may make a poor class prototype vector. For example, we used such "prototypes" for ranking instances for each class in Reuters-21578 and newsgroup. The ranking quality (max F1) was significantly lower than that obtained from a single perceptron or a linear SVM trained for the class (5 to 10% reduction in absolute value of Max-F1 compared to perceptron on Reuters-21578 classes). On the newsgroup data set, the Max-F1 performances were comparable to single perceptrons but lagged the performance of SVMs.

6.4.4 CLASS INDEGREES

In Section 4.4 it was mentioned that prototypes may require more (nonzero) weights and processing time than features, and thus feature-based methods could have an efficiency advantage over

prototype-based methods (even when adjusted for the average vector length times average feature outdegree). Of course, this all depends on the details of what processing needs to be performed for a given algorithm and what the average numbers come out to. It may be useful to look at the average indegree of a class during the FF algorithm on our data sets.

Let the indegree of a class, that is, the length of the prototype vector, be the number of features that have a significant edge to the class (within the highest d_{max} edges for each feature). After one pass of training, the indegree for the top ranking class (averaged over test instances), for the Newsgroup, Industry, RCV1, Ads, and Web was respectively: 6k, 2k, 4k, 530, and 14k. The true class had a lower but somewhat similar average indegree, except for the web, where the true class had an average indegree of 6700. Furthermore, in general, the average indegree of classes at given rank goes down with increasing rank. This is plausible: concepts with relatively higher indegree (i.e., more connections) tend to beat others in the score received: they tend to be ranked closer to the top.

Observe that the uniform averages (indegree of a class picked uniformly at random) is significantly lower for the big data sets, due to the skew in class frequencies. The uniform averages can be computed from Figures 9 and 10, for example, for the Web data it is: $\frac{1.5meg}{14k} \approx 100$.

6.4.5 EXAMPLE FEATURES AND CONNECTIONS

On RCV1, there were about 300 feature-class connections with weight greater than 0.9 (strong connections). Examples included: "figurehead" to the class "EQUITY MARKETS", "gunfir" to the class "WAR, CIVIL WAR", and "manuf" (manufacturing) to "LEADING INDICATORS". Examples of features with relatively large "leaks", that is, with $w_f = \sum_c w_{f,c} < 0.25$, and thus likely uninformative, included "ago", "base", "year", and "intern".[21]

7. Conclusions

We raised the challenge of large-scale many-class learning and explored the approach of index learning. In this index-learning context, we began with the informal conjecture that (1) each feature need only connect to a relatively small number of classes, and (2) these connections can be discovered efficiently. We provided evidence that there exist very efficient online learning algorithms that nevertheless enjoy competitive and at times better accuracy performance than other commonly used methods. The algorithms may best be viewed as performing the computations from the side of features (the predictors) rather than the classes (the predicted). Each feature computes that choice of classes it may connect to and the connection weights. In particular, for very large-scale problems, each feature is space constrained in performing its computations and in the number of classes to which it can connect.

Much work remains in terms of advancing the algorithms and developing an understanding of their successes and limitations, including developing insights into the possible regularities in naturally occurring data that could explain the observed successes. We intend to further investigate index-learning algorithms, including different update methods and objectives, to develop theoretical properties, and to explore applications to various domains.

21. The feature "the" was probably dropped (a "stop" word) during the tokenization of this data set (Lewis et al., 2004).

Acknowledgments

Thanks to the anonymous reviewers of the paper for their valuable feedback and suggestions, which improved the presentation. Thanks to Scott Corlett, Dennis DeCoste, Scott Gaffney, Jian Huang, Sathiya Keerthy, David Kempe, Ravi Kumar, John Langford, Chih-Jen Lin, Kishore Papineni, Hema Raghavan, Lance Riedel, Mohammad Salavatipour, and the machine learning group at Yahoo! Research for suggestions, pointers, and discussions, and Pradhuman Jahala, Xiaofei He, and Cliff Brunk for providing us the web data.

Appendix A. NP-Hardness

For the purpose of establishing hardness, the problem is specified by a finite set of instances, wherein each instance is assigned a class and specified by the set of its active features. The features need only be Boolean. Of course, more general problems are at least as hard. We show NP-hardness when a fixed upper constraint is imposed on the outdegree on each feature in the index. The problem is NP-hard under either objective of maximizing accuracy or maximizing the MRR reward on the given set. For MRR, for each instance, the reward is the reciprocal rank $\frac{1}{k_x}$, that is, the rank of the correct class in the ranking returned by the index. On a single instance, the reward could be 0, if the class is not retrieved, and maxes at 1, if the correct class has rank 1. Note that MRR in Section 3.3 is simply the average reward per instance. For accuracy (R_1), the reward is either 1, if the correct class is ranked highest, or otherwise 0. The decision problem is then to determine whether a weighted index (a weighted bipartite graph) satisfying the outdegree constraint exists that yields a total reward, $\sum_{x \in X} r(c_x)$, exceeding a desired threshold.

Theorem 4 *The index learning problem with the objective of either maximizing accuracy (R_1) or minimizing HR on a given set of instances, with the constraint of a constant upper bound, such as 1, on the outdegree of each feature is NP-Hard.*

Proof The reduction is from the SET COVER problem (Garey and Johnson., 1979). We reduce the SET COVER problem to problem of computing an index wherein each feature can connect to at most 1 class.

An instance I of SET COVER consist of a set $U = \{e_1, \ldots, e_n\}$ of elements and a set $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of U. The goal is to find a smallest subset $\mathcal{S}' \subseteq S$ such that $\bigcup_{S_i \in \mathcal{S}'} = U$. Given a SET COVER instance I, we construct an instance of the indexing problem with only two classes c_1 and c_2 such that there is a SET COVER solution of size C for I iff there is an index (with the maximum outdegree of 1 constraint), such that the maximum total reward, the number of instances for which the right class is ranked highest, is $|U| + |\mathcal{S}| - C$.

In the constructed indexing problem, there is one feature f_i corresponding to each set $S_i \in \mathcal{S}$, for a total of m features. There is also one instance x_j for each element $e_j \in U$ ($1 \leq j \leq n$), and x_j contains feature f_i (x_j is connected to f_i) iff the element e_j belongs to the set S_i. These instances, called the "original instances", belong to class c_1. In addition, there are m "extra" instances, one for each set (or each feature). Each of these extra instances contains only the feature it corresponds to, and belongs only to class c_2 (see Figure 20).

Here, in constructing an index, we need to decide for each feature whether to connect the feature to c_1 or to c_2 (we can only connect to one of the two), and with what weights. Now, if a cover of size C exists, then we can easily obtain an index yielding reward of $|U| + |\mathcal{S}|$ - C: we connect the

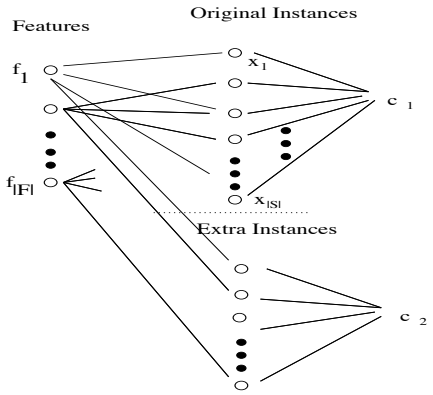


Figure 20: Reduction of the minimum set cover problem to index learning.

features in the cover (i.e., those features whose corresponding sets are in the cover) to c_1, each with weight of $|\mathcal{S}|$, and we connect all the other features to c_2 with a relatively small weight of say 1. In this way, for any original instance ($|U|$ many), c_1 is ranked highest, as at least one of its feature (the one(s) in the cover) connects to c_1 with high weight. For only $|\mathcal{S}|-C$ many of the extra instances, the correct class is missed, thus the total reward is $|U|+|\mathcal{S}|-C$.

For the reverse direction, we want to show that if an index with reward R exists, then there is a cover size $C \leq |U|+|\mathcal{S}|-R$. Assume an index is given with reward R. Note that lowering the connection weights to c_2 does not degrade the reward. So assume all such weights are at fixed minimum value v_{min}. Next, we note that any index can be converted to one in which all the original instances are "covered", that is, the index ranks the right class highest: take any original instance for which this is not the case, and take one of its features that is connected to c_2 (there must be at least one), drop that edge, and connect it to c_1 with high enough weight so that c_1 is ranked highest. The weight can simply be $v_{min}|\mathcal{S}|$. This operation does not degrade total reward as we lose on exactly one extra instance, but gain on at least one original instance. We may repeat this operation until all original instances are covered, and the reward is now $R' \geq R$. Now, we see that $R' = |U|+|\mathcal{S}|-n$, where n is the number of those extra instances for which c_2 is not retrieved, equal to the number of features covering the original instances (connecting to c_1), or the cover size in the original problem is $C = n = |U|+|\mathcal{S}|-R' \leq |U|+|\mathcal{S}|-R$. ■

Observe that the NP-hardness remains and is easier to show if we use the maximum incoming score rule for class retrieval (each class gets the maximum of its incoming edge weights) instead of the sum. This reduction does not establish NP-hardness of constant-ratio approximability of class ranking (due to the subtraction), which remains an interesting open problem. For instance, either a constant-ratio approximation to loss (for problems with high accuracy) or accuracy (for problems with high loss) would be interesting. A similar reduction for the problem of computing an *unweighted* index shows that problem is NP-hard even to approximate (Madani et al., 2007).

Appendix B. Approximation Consequences of Edge Dropping

Consider the setting of Section 4.1 wherein a feature wants to compute the proportions of the (sufficiently frequent) classes in the stream it observes. There are two causes for inaccuracies in computing proportions:

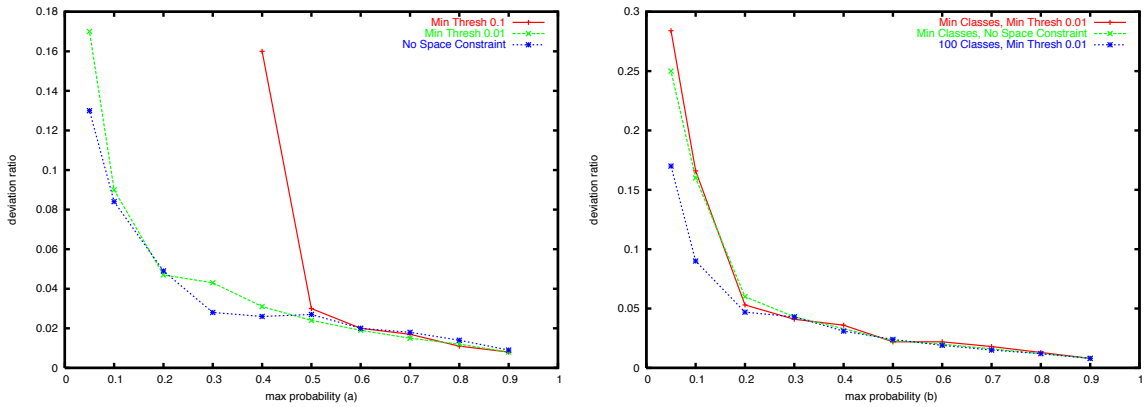


Figure 21: The performance of FSU under different allowances w_{min}. FSU computes the class proportions from processing a stream of 1000 class observations. For a choice of highest true probability p^*, the remaining probability mass $(1-p^*)$ is spread evenly over remaining classes. This is done under two regimes of generating classes. In (a) the number of unique classes is fixed at $|\mathcal{C}| = 100$, and in (b) it is $|\mathcal{C}| = \lceil \frac{1}{p^*} \rceil$ (i.e., modeling the situation in which other classes tie or have close probabilities to the maximum). The experiment, consisting of picking a class distribution and generating a 1000 draws, is repeated for 200 trials. In each trial, the deviation ratio of the highest proportion value computed by FSU, $\tilde{p}_{c_1}$, from the true maximum probability, p^*, is computed. This deviation (ratio) is $\frac{|\tilde{p}_{c_1} - p^*|}{p^*}$. The average deviation over the 200 trials is plotted against p^*. In the plot of part (b), the deviation is also compared to the case of 100 classes and $w_{min} = 0.01$. We note that $w_{min} \approx 0.01$ appears satisfactory for $p^* \geq 0.05$, while $w_{min} \approx 0.1$ performs well for a much smaller range.

- Finite samples (at any given time only a finite sample has been observed).
- Setting small weights (below w_{min}) to 0 (dropping edges) to save memory.

As FSU may drop and reinsert edges repeatedly, its approximation of actual proportions suffers from more than the issue of finite samples alone. We want to get an idea of this extra loss that we incur compared to the case when memory is not an issue (when no edges are dropped). Intuitively, FSU should work well as long as the proportions we are interested in sufficiently exceed the w_{min} threshold. The probability that a class with say probability p is not seen in some $\frac{1}{w_{min}}$ trials is $(1-p)^{1/w_{min}}$, and as long the ratio $\frac{p}{w_{min}}$ is high (several multiples), for example, $p > 4w_{min}$, this probability is relatively small. For example, for $w_{min} = 0.01$, and $p = 0.05$, the probability of not seeing such a class for a stretch of 100 consecutive trials is 0.006. More generally, the chance of being set to 0 (dropped) for a class with occurrence probability p quickly diminishes as we increase the ratio $\frac{p}{w_{min}}$, and therefore the cause of inaccuracies due to finite memory (the outdegree constraint on features) is mitigated.

We conducted experiments to see how much the proportion estimation by FSU deviates from true proportions and in particular compared that deviation to the deviations when FSU is not memory constrained (when w_{min} is set to 0). Figures 21 and 22 show the results. The experiments differ on how we generated the classes and computed the deviations. In the first of these experiments, to generate the true-class distribution, for some fixed number of classes $|\mathcal{C}|$, one class is given a

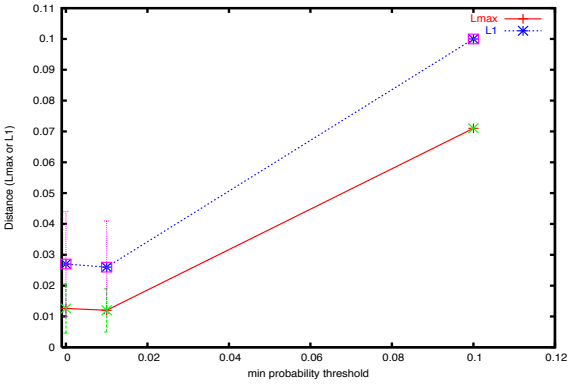


Figure 22: The performance of FSU, under different allowances w_{min}. The vector of true class probabilities is generated by uniformly and sequentially picking from $[0,1]$, and keeping track of total mass p (which should not exceed 1). If for latest generated class the probability drawn is greater than remaining mass $1-p$, the remaining mass is assigned instead, and class generation is stopped. FSU is evaluated after seeing a stream of 1000 classes iid drawn from such a source. The l_1 and l_∞ (or l_{max}) distances between the vector of empirical proportions that FSU computes and the true probabilities vector, averaged over 200 trials, is reported. FSU with w_{min} =0.01 yields distances comparable to FSU with $w_{min} = 0$, but w_{min} =0.1 yields significantly inferior estimates.

highest probability p^*, and the remaining classes obtain the remaining probability mass divided uniformly: $\frac{1-p^*}{|\mathcal{C}|-1}$. We then generated a stream of 1000 class observations (1000 iid draws) from such a distribution, and gave it to FSU with different values of w_{min}. We computed the deviation ratio: $\frac{|\tilde{p}_{c_1}-p^*|}{p^*}$, where c_1 denotes the class ranked highest by FSU, and $\tilde{p}_{c_1}$ is its assigned probability (highest computed probability). We averaged this deviation (ratio) over 200 trials of repeating the experiments. Figure 21(a) shows the averages when $|\mathcal{C}| = 100$ (so all classes except for one, obtain $\frac{1-p^*}{99}$). Figure 21(b) shows the results for $|\mathcal{C}| = \lceil \frac{1}{p^*} \rceil$ (e.g., for when $p^* \geq 0.5$, $|\mathcal{C}| = 2$, and when $p^* = 0.05, |\mathcal{C}| = 20$). Thus Figure 21(b) shows how FSU with limited w_{min} compares when the classes have similar proportions.

In the second set of experiments, we generated the probability for each class uniformly from the $[0,1]$ interval, keeping track of the total probability p used up during the course of generation. If the newest class gets a probability greater than $1-p$, $1-p$ is assigned to it and class generation, for selecting a distribution, is stopped. We then sampled iid to get a sequence of 1000 class observations. We compared the vector of class proportions that FSU computed using l_1 or l_∞ distance against the vector of true probabilities. We averaged the distances over 200 trials. We plot the results for FSU under different w_{min} constraints. We see that a threshold of $w_{min} \geq 0.1$ is not appropriate if the proportions we are interested in may be below 0.5, but a threshold of $w_{min} \approx 0.01$ does well, if we are interested in true proportions that are greater than 0.05 say. We compared a number of other statistics, such as the maximum deviation from true probability, and the probability that the deviation is larger than a threshold, and FSU with $w_{min} = 0.01$ performed similarly to $w_{min} = 0$ on the distributions tested. The reason as alluded to earlier is that those classes with proportions

significantly greater than w_{min} have a high chance of being seen early and frequently enough in the stream and not being dropped.

Thus, as long as we expect that the useful proportions are a few multiples away from the w_{min} we choose, FSU is expected to compute proportions that are close to ones computed by the FSU with w_{min} set to 0 (no space constraints). Further, we expected that most often the important feature connection weights that determine the true classes during ranking have fairly high weight. Note also that the constraint of finite samples also points to the limited utility of trying to keep track of relatively low proportions: for most useful features, we may see them below say a 1000 times (in common data sets), and commonly occurring features tend not to be discriminative. Finally, vector length is a factor: if there tend to exist strong features-class connections, the influence of the weaker connections on changing the ranking will be limited, in particular when the number of active features is adequately small. Thus, in many practical learning problems, expecting that most useful proportions (weights) are in a relatively small interval, say $[0.05, 1]$ (or that the features do not require high outdegree) may be reasonable (see Section 6.3.3). In general however, some experimentation may be required to set the w_{min} parameter.

References

S. Albers and J. Westbrook. Self-organizing data structures. In A. Fiat and G. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 31–51. Springer LNCS 1442, 1998.

J. K. Anlauf and M. Biehl. The adatron: an adaptive perceptron algorithm. *Europhysics Letters*, 1989.

H. Aradhye, G. Toderici, and J. Yagnik. Video2text: Learning to annotate video content. In *IEEE Int. Conf. on Data Mining (ICDM) Workshop on Internet Multimedia Mining*, 2009.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

R. Bayardo, Y. Ma, and R. Srikant. Scaling up all-pairs similarity search. In *Proc. Int. World Wide Web Conference (WWW)*, 2007.

A. Blum. Empirical support for winnow and wighted majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26:5–23, 1997.

A. Borodin and R. El Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

V. R. Carvalho and W. Cohen. Single pass online learning. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2006.

N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. Schapire, and M. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, 1997.

T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, 2009.

W. W. Cohen and Y. Singer. Context-senstive learning methods for text categorization. *ACM Trans. on Information Systems (TOIS)*, 17:141–173, 1999.

K. Crammer and Y. Singer. A new family of online algorithms for category ranking. *Journal of Machine Learning Research (JMLR)*, 3:1025–1058, 2003a.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003b.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

O. Dekel, J. Keshet, and Y. Singer. Large margin hierarchical classification. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2003.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

S. Dumais and H. Chen. Hierarchical classification of web content. In *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, 2000.

Y. Even-Zohar and D. Roth. A classification approach to word prediction. In *Proc. of the 1st North Amercian Association of Computational Linguistics (NAACL)*, 2000.

M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. Ullman. Computing iceberg queries efficiently. In *Proc. 24th Int. Conf. Very Large Scale Data Bases (VLDB)*, 1998.

S. Fidler and A. Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *Proc. of IEEE Int. Conf. on Vision and Pattern Recognition (CVPR)*, 2007.

D. A. Forsyth and J. Ponce. *Computer Vision*. Prentice Hall, 2003.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

Y. Freund, R. Schapire, Y. Singer, and M. Warmuth. Using and combining predictors that specialize. In *Proc. ACM Symposum on Theory of Computing (STOC)*, 1997.

E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipida-baed explicit semantic analysis. In *Proc. Int. Joint Conf. on AI (IJCAI)*, 2007.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

C. Genest and J. V. Zidek. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1(1):114–148, 1986.

C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

P. B. Gibbons and Y. Matias. Synopsis data structures for massive data sets. In *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science: Special Issue on Eternal Memory Algorithms and Visualization*, 1999.

J. T. Goodman. A bit of progress in language modeling. *Computer Speech and Language*, 15(4): 403–434, October 2001.

K. Grill-Spector and N. Kanwisher. Visual recognition, as soon as you know it is there, you know what it is. *Pscychological Science*, 16(2):152–160, 2005.

M. Grobelnik and D. Mladenic. Efficient text categorization. In *Text Mining Workshop at European Conf. on Machine Learning (ECML)*. 1998.

S. Guha and A. McGregor. Space-efficient sampling. In *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2007.

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

C. J. Hsieh, K. J. Chang, C. J. Lin, and S. Sathiya Keerthi. A dual coordinate descent method for large-scale linear SVM. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2008.

J. Huang, O. Madani, and C. Lee Giles. Error-driven generalist+experts (EDGE): A multi-stage ensemble framework for text categorization. In *Proc. ACM Conf. on Information and Knowledge Management (CIKM)*, 2008.

R. M. Karp, C. H. Papadimitriou, and S. Shenker. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. Database Systems (TODS)*, 28:51–55, 2003.

S. Keerthi and D. DeCoste. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research (JMLR)*, 6:341–361, 2005.

D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proc. Int. Conf. on Machine Learning (ICML)*, 1997.

W. Krauth and M. Mezard. Learning algorithms with optimal stability in neural networks. *J. of Physics A*, 20, 1987.

K. Lang. Newsweeder: Learning to filter netnews. In *Proc. Int. Conf. on Machine Learning*, 1995.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research (JMLR)*, 5:361–397, 2004.

Y. Li and P. M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1-3), 2002.

Y. Li, H. Zaragoza, R. Herbrich, J. Shawe-Taylor, and J. Kandola. The perceptron algorithm with uneven margins. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2002.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

T. Liu, Y. Yang, H. Wan, H. Zeng, Z. Chen, and W. Ma. Support vector machines classification with very large scale taxonomy. *SIGKDD Explorations*, 7, 2005.

O. Madani. Exploring massive learning via a prediction system. In *AAAI Fall Symposium Series: Computational Approaches to Representation Change During Learning and Development*, 2007a.

O. Madani. Prediction games in infinitely rich worlds. Technical Report 2, Yahoo! Research (and workshop on Utility Based Data Mining (UBDM)'06), June 2007b.

O. Madani and M. Connor. Ranked Recall: Efficient classification by efficient learning of indices that rank. Technical Report 3, Yahoo! Research, 2007.

O. Madani and M. Connor. Large-scale many-class learning. In *SIAM Conf. on Data Mining (SDM)*, 2008.

O. Madani and J. Huang. On updates that constrain the features' connections during learning. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2008.

O. Madani, W. Greiner, D. Kempe, and M. Salavatipour. Recall systems: Efficient learning and use of category indices. In *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2007.

O. Madani, H. Bui, and E. Yeh. Efficient online learning and prediction of users' desktop actions. In *Proc. Int. Joint Conf. on AI (IJCAI)*, 2009.

C. Mesterharm. A multiclass linear learning algorithm related to Winnow. In *Proc. Neural Information Processing Systems (NIPS)*, 2000.

C. Mesterharm. Transforming linear-threshold learning algorithms into multiclass linear learning algorithms. Technical Report dcs-tr-460, Rutgers, 2001.

G. L. Murphy. *The Big Book of Concepts*. MIT Press, 2002.

J. Platt. Probabilities for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schlkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

D. R. Radev, H. Qi, H. Wu, and W. Fan. Evaluating web-based question answering systems. In *Proc. Int. Conf. on Language Resources and Evaluation (LREC)*, 2002.

H. Raghavan, O. Madani, and R. Jones. When will a human in the loop accelerate learning? quantifying the complexity of classification problems. In *Int. Workshop on AI for Human Computing, at IJCAI*, 2007.

J. Rennie, L. Shih, J. Teevan, and D. Karger. Tackling the poor assumption of Naive Bayes text classifiers. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2003.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research (JMLR)*, 5, 2004.

T. G. Rose, M. Stevenson, and Miles Whitehead. The reuters corpus vol. 1 - from yesterday's news to tomorrow's language resources. In *Proc. Int. Conf. on Lang. Resources and Evaluation (LREC)*, 2002.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34: 1–47, 2002.

S. Shalev-Schwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2007.

S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381: 520–522, 1996.

T. Turtle and J. Flood. Query evaluation: Strategies and optimizations. *Information Processing & Management*, 31(6), 1995.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 2000.

V. G. Vovk. Aggregating strategies. In *Annual Workshop on Computational Learning Theory*, 1990.

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, 2001.

I. H. Witten, T. C. Bell, and A. Moffat. *Managing Gigabytes: Compressing and Indexing Documents and Images*. John Wiley & Sons, 1994.

GR. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, 2008.

Journal of Machine Learning Research 10 (2009) 2615-2637 Submitted 2/09; Revised 8/09; Published 11/09

Hash Kernels for Structured Data

Qinfeng Shi QINFENG.SHI@GMAIL.COM
College of Engineering and Computer Science
The Australian National University
Canberra, ACT 0200, Australia

James Petterson JPETTERSON@GMAIL.COM
Statistical Machine Learning
National ICT Australia
Locked Bag 8001
Canberra, ACT 2601, Australia

Gideon Dror GIDEON@MTA.AC.IL
Division of Computer Science
Academic College of Tel-Aviv-Yaffo, Israel

John Langford JL@HUNCH.NET
Alex Smola ALEX.SMOLA@GMAIL.COM
Yahoo! Research
New York, NY and Santa Clara, CA, USA

S.V.N. Vishwanathan VISHY@MAIL.RSISE.ANU.EDU.AU
Department of Statistics
Purdue University, IN, USA

Editor: Soeren Sonnenburg, Vojtech Franc, Elad Yom-Tov and Michele Sebag

Abstract

We propose hashing to facilitate efficient kernels. This generalizes previous work using sampling and we show a principled way to compute the kernel matrix for data streams and sparse feature spaces. Moreover, we give deviation bounds from the exact kernel matrix. This has applications to estimation on strings and graphs.

Keywords: hashing, stream, string kernel, graphlet kernel, multiclass classification

1. Introduction

In recent years, a number of methods have been proposed to deal with the fact that kernel methods have slow runtime performance if the number of kernel functions used in the expansion is large. We denote by $\mathcal{X}$ the domain of observations and we assume that $\mathcal{H}$ is a Reproducing Kernel Hilbert Space $\mathcal{H}$ with kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

1.1 Keeping the Kernel Expansion Small

One line of research (Burges and Schölkopf, 1997) aims to reduce the number of basis functions needed in the overall function expansion. This led to a number of reduced set Support Vector algorithms which work as follows: a) solve the full estimation problem resulting in a kernel expansion,

b) use a subset of basis functions to approximate the exact solution, c) use the latter for estimation. While the approximation of the full function expansion is typically not very accurate, very good generalization performance is reported. The big problem in this approach is that the optimization of the reduced set of vectors is rather nontrivial.

Work on estimation on a budget (Dekel et al., 2006) tries to ensure that this problem does not arise in the first place by ensuring that the number of kernel functions used in the expansion never exceeds a given budget or by using an ℓ_1 penalty (Mangasarian, 1998). For some algorithms, for example, binary classification, guarantees are available in the online setting.

1.2 Keeping the Kernel Simple

A second line of research uses variants of sampling to achieve a similar goal. That is, one uses the feature map representation

$$k(x,x') = \left\langle \phi(x), \phi(x') \right\rangle.$$

Here ϕ maps $\mathcal{X}$ into some feature space $\mathcal{F}$. This expansion is approximated by a mapping $\overline{\phi} : \mathcal{X} \to \overline{\mathcal{F}}$

$$\overline{k}(x,x') = \left\langle \overline{\phi}(x), \overline{\phi}(x') \right\rangle \text{ often } \overline{\phi}(x) = M\phi(x).$$

Here $\overline{\phi}$ has more desirable computational properties than ϕ. For instance, $\overline{\phi}$ is finite dimensional (Fine and Scheinberg, 2001; Kontorovich, 2007; Rahimi and Recht, 2008), or $\overline{\phi}$ is particularly sparse (Li et al., 2007).

1.3 Our Contribution

Firstly, we show that the sampling schemes of Kontorovich (2007) and Rahimi and Recht (2008) can be applied to a considerably larger class of kernels than originally suggested—the authors only consider languages and radial basis functions respectively. Secondly, we propose a biased approximation $\overline{\phi}$ of ϕ which allows efficient computations even on data streams. Our work is inspired by the count-min sketch of Cormode and Muthukrishnan (2004), which uses hash functions as a computationally efficient means of randomization. This affords storage efficiency (we need not store random vectors) and at the same time they give performance guarantees comparable to those obtained by means of random projections.

As an application, we demonstrate computational benefits over suffix array string kernels in the case of document analysis and we discuss a kernel between graphs which only becomes computationally feasible by means of compressed representation.

1.4 Outline

We begin with a description of previous work in Section 2 and propose the hash kernels in Section 3 which is suitable for data with simple structure such as strings. An analysis follows in Section 4. And we propose a graphlet kernel which generalizes hash kernels to data with general structure—graphs and discuss a MCMC sampler in Section 5. Finally we conclude with experiments in Section 6.

2. Previous Work and Applications

Recently much attention has been paid to efficient algorithms with randomization or hashing in the machine learning community.

2.1 Generic Randomization

Kontorovich (2007) and Rahimi and Recht (2008) independently propose the following: denote by $c \in \mathcal{C}$ a random variable with measure P. Moreover, let $\phi_c : \mathcal{X} \to \mathbb{R}$ be functions indexed by $c \in \mathcal{C}$. For kernels of type

$$k(x,x') = \mathbf{E}_{c\sim \mathrm{P}(c)} \left[\phi_c(x)\phi_c(x')\right] \tag{1}$$

an approximation can be obtained by sampling $C = \{c_1,\ldots,c_n\} \sim \mathrm{P}$ and expanding

$$\overline{k}(x,x') = \frac{1}{n}\sum_{i=1}^{n} \phi_{c_i}(x)\phi_{c_i}(x').$$

In other words, we approximate the feature map $\phi(x)$ by $\overline{\phi}(x) = n^{-\frac{1}{2}}\left(\phi_{c_1}(x),\ldots,\phi_{c_n}(x)\right)$ in the sense that their resulting kernel is similar. Assuming that $\phi_c(x)\phi_c(x')$ has bounded range, that is, $\phi_c(x)\phi_c(x') \in [a,a+R]$ for all c, x and x' one may use Chernoff bounds to give guarantees for large deviations between $k(x,x')$ and $\overline{k}(x,x')$. For matrices of size $m \times m$ one obtains bounds of type $O(R^2\varepsilon^{-2}\log m)$ by combining Hoeffding's theorem with a union bound argument over the $O(m^2)$ different elements of the kernel matrix. The strategy has widespread applications beyond those of Kontorovich (2007) and Rahimi and Recht (2008):

- Kontorovich (2007) uses it to design kernels on regular languages by sampling from the class of languages.
- The marginalized kernels of Tsuda et al. (2002) use a setting identical to (1) as the basis for comparisons between strings and graphs by defining a random walk as the feature extractor. Instead of exact computation we could do sampling.
- The Binet-Cauchy kernels of Vishwanathan et al. (2007b) use this approach to compare trajectories of dynamical systems. Here c is the (discrete or continuous) time and $\mathrm{P}(c)$ discounts over future events.
- The empirical kernel map of Schölkopf (1997) uses $\mathcal{C} = \mathcal{X}$ and employs some kernel function κ to define $\phi_c(x) = \kappa(c,x)$. Moreover, $\mathrm{P}(c) = \mathrm{P}(x)$, that is, placing our sampling points c_i on training data.
- For RBF kernels, Rahimi and Recht (2008) use the fact that k may be expressed in the system of eigenfunctions which commute with the translation operator, that is the Fourier basis

  $$k(x,x') = \mathbf{E}_{w\sim \mathrm{P}(w)}[e^{-i\langle w,x\rangle} e^{i\langle w,x'\rangle}]. \tag{2}$$

  Here $\mathrm{P}(w)$ is a nonnegative measure which exists for any RBF kernel by virtue of Bochner's theorem, hence (2) can be recast as a special case of (1). What sets it apart is the fact that the variance of the features $\phi_w(x) = e^{i\langle w,x\rangle}$ is relatively evenly spread. (2) extends immediately to Fourier transformations on other symmetry groups (Berg et al., 1984).
- The conditional independence kernel of Watkins (2000) is one of the first instances of (1). Here $\mathcal{X},\mathcal{C}$ are domains of biological sequences, $\phi_c(x) = \mathrm{P}(x|c)$ denotes the probability of observing x given the ancestor c, and $\mathrm{P}(c)$ denotes a distribution over ancestors.

While in many cases straightforward sampling may suffice, it can prove disastrous whenever $\phi_c(x)$ has only a small number of significant terms. For instance, for the pair-HMM kernel most strings c are *unlikely* ancestors of x and x', hence $P(x|c)$ and $P(x'|c)$ will be negligible for most c. As a consequence the number of strings required to obtain a good estimate is prohibitively large—we need to reduce $\overline{\phi}$ further.

2.2 Locally Sensitive Hashing

The basic idea of randomized projections (Indyk and Motawani, 1998) is that due to concentration of measures the inner product $\langle \phi(x), \phi(x') \rangle$ can be approximated by $\sum_{i=1}^{n} \langle v_i, \phi(x) \rangle \langle v_i, \phi(x') \rangle$ efficiently, provided that the distribution generating the vectors v_i satisfies basic regularity conditions. For example, $v_i \sim \mathcal{N}(0, I)$ is sufficient, where I is an identity matrix. This allows one to obtain Chernoff bounds and $O(\varepsilon^{-2} \log m)$ rates of approximation, where m is the number of instances. The main cost is to store v_i and perform the $O(nm)$ multiply-adds, thus rendering this approach too expensive as a preprocessing step in many applications.

Achlioptas (2003) proposes a random projection approach that uses symmetric random variables to project the original feature onto a lower dimension feature space. This operation is simple and faster and the author shows it does not sacrifice the quality of the embedding. Moreover, it can be directly applied to online learning tasks. Unfortunately, the projection remains dense resulting in relatively poor computational and space performance in our experiments.

2.3 Sparsification

Li et al. (2007) propose to sparsify $\phi(x)$ by randomization while retaining the inner products. One problem with this approach is that when performing optimization for linear function classes, the weight vector w which is a linear combination of $\phi(x_i)$ remains large and dense, thus obliterating a significant part of the computational savings gained in sparsifying ϕ.

2.4 Count-Min Sketch

Cormode and Muthukrishnan (2004) propose an ingenious method for representing data streams. Denote by $\mathcal{I}$ an index set. Moreover, let $h : \mathcal{I} \to \{1, \ldots, n\}$ be a hash function and assume that there exists a distribution over h such that they are pairwise independent.

Assume that we draw d hash functions h_i at random and let $S \in \mathbb{R}^{n \times d}$ be a sketch matrix. For a stream of symbols s update $S_{h_i(s),i} \leftarrow S_{h_i(s),i} + 1$ for all $1 \leq i \leq d$. To retrieve the (approximate) counts for symbol s' compute $\min_i S_{h_i(s'),i}$. Hence the name count-min sketch. The idea is that by storing counts of s according to several hash functions we can reduce the probability of collision with another particularly large symbol. Cormode and Muthukrishnan (2004) show that only $O(\varepsilon^{-1} \log 1/\delta)$ storage is required for an ε-good approximation, where $1 - \delta$ is the confidence.

Cormode and Muthukrishnan (2004) discuss approximating inner products and the extension to signed rather than nonnegative counts. However, the bounds degrade for real-valued entries. Even worse, for the hashing to work, one needs to take the minimum over a set of inner product candidates.

2.5 Random Feature Mixing

Ganchev and Dredze (2008) provide empirical evidence that using hashing can eliminate alphabet storage and reduce the number of parameters without severely impacting model performance. In addition, Langford et al. (2007) released the Vowpal Wabbit fast online learning software which uses a hash representation similar to the one discussed here.

2.6 Hash Kernel on Strings

Shi et al. (2009) propose a hash kernel to deal with the issue of computational efficiency by a very simple algorithm: high-dimensional vectors are compressed by adding up all coordinates which have the same hash value—one only needs to perform as many calculations as there are nonzero terms in the vector. The hash kernel can jointly hash both label and features, thus the memory footprint is essentially independent of the number of classes used.

3. Hash Kernels

Our goal is to design a possibly biased approximation which a) approximately preserves the inner product, b) which is generally applicable, c) which can work on data streams, and d) which increases the density of the feature matrices (the latter matters for fast linear algebra on CPUs and graphics cards).

3.1 Kernel Approximation

As before denote by $\mathcal{I}$ an index set and let $h : \mathcal{I} \to \{1,\dots,n\}$ be a hash function drawn from a distribution of pairwise independent hash functions. Finally, assume that $\phi(x)$ is indexed by $\mathcal{I}$ and that we may compute $\phi_i(x)$ for all nonzero terms efficiently. In this case we define the hash kernel as follows:

$$\overline{k}(x,x') = \left\langle \overline{\phi}(x), \overline{\phi}(x') \right\rangle \text{ with } \overline{\phi}_j(x) = \sum_{i \in \mathcal{I}; h(i)=j} \phi_i(x) \tag{3}$$

We are accumulating all coordinates i of $\phi(x)$ for which $h(i)$ generates the same value j into coordinate $\overline{\phi}_j(x)$. Our claim is that hashing preserves information as well as randomized projections with significantly less computation. Before providing an analysis let us discuss two key applications: efficient hashing of kernels on strings and cases where the number of classes is very high, such as categorization in an ontology.

3.2 Strings

Denote by $\mathcal{X} = \mathcal{I}$ the domain of strings on some alphabet. Moreover, assume that $\phi_i(x) := \lambda_i \#_i(x)$ denotes the number of times the substring i occurs in x, weighted by some coefficient $\lambda_i \geq 0$. This allows us to compute a large family of kernels via

$$k(x,x') = \sum_{i \in \mathcal{I}} \lambda_i^2 \#_i(x) \#_i(x'). \tag{4}$$

Teo and Vishwanathan (2006) propose a storage efficient $O(|x| + |x'|)$ time algorithm for computing k for a *given* pair of strings x, x'. Here $|x|$ denotes the length of the string. Moreover, a weighted combination $\sum_i \alpha_i k(x_i, x)$ can be computed in $O(|x|)$ time after $O(\sum_i |x_i|)$ preprocessing.

The big drawback with string kernels using suffix arrays/trees is that they require large amounts of working memory. Approximately a factor of 50 additional storage is required for processing and analysis. Moreover, updates to a weighted combination are costly. This makes it virtually impossible to apply (4) to millions of documents. Even for modest document lengths this would require Terabytes of RAM.

Hashing allows us to reduce the dimensionality. Since for every document x only a relatively small number of terms $\#_i(x)$ will have nonzero values—at most $O(|x|^2)$ but in practice we will restrict ourselves to substrings of a bounded length l leading to a cost of $O(|x| \cdot l)$—this can be done efficiently in a single pass over x. Moreover, we can compute $\overline{\phi}(x)$ as a pre-processing step and discard x altogether.

Note that this process spreads out the features available in a document *evenly* over the coordinates of $\overline{\phi}(x)$. Moreover, note that a similar procedure can be used to obtain good estimates for a TF/IDF reweighting of the counts obtained, thus rendering preprocessing as memory efficient as the actual computation of the kernel.

3.3 Multiclass

Classification can sometimes lead to a very high dimensional feature vector even if the underlying feature map $x \to \phi(x)$ may be acceptable. For instance, for a bag-of-words representation of documents with 10^4 unique words and 10^3 classes this involves up to 10^7 coefficients to store the parameter vector directly when the $\phi(x,y) = e_y \otimes \phi(x)$, where $\otimes$ is the tensor product and e_y is a vector whose y-th entry is 1 and the rest are zero. The dimensionality of e_y is the number of classes.

Note that in the above case $\phi(x,y)$ corresponds to a sparse vector which has nonzero terms only in the part corresponding to e_y. That is, by using the joint index (i,y) with $\phi(x,y)_{(i,y')} = \phi_i(x)\delta_{y,y'}$ we may simply apply (3) to an extended index to obtain hashed versions of multiclass vectors. We have

$$\overline{\phi}_j(x,y) = \sum_{i \in \mathbb{J}; h(i,y)=j} \phi_i(x).$$

In some cases it may be desirable to compute a compressed version of $\phi(x)$, that is, $\overline{\phi}(x)$ first and subsequently expand terms with y. In particular for strings this can be useful since it means that we need not parse x for every potential value of y. While this deteriorates the approximation in an additive fashion it can offer significant computational savings since all we need to do is permute a given feature vector as opposed to performing any summations.

3.4 Streams

Some features of observations arrive as a stream. For instance, when performing estimation on graphs, we may obtain properties of the graph by using an MCMC sampler. The advantage is that we need not store the entire data stream but rather just use summary statistics obtained by hashing.

4. Analysis

We show that the penalty we incur from using hashing to compress the number of coordinates only grows *logarithmically* with the number of objects and with the number of classes. While we are

unable to obtain the excellent $O(\varepsilon^{-1})$ rates offered by the count-min sketch, our approach retains the inner product property thus making hashing accessible to linear estimation.

4.1 Bias and Variance

A first step in our analysis is to compute bias and variance of the approximation $\overline{\phi}(x)$ of $\phi(x)$. Whenever needed we will write $\overline{\phi}^h(x)$ and $\overline{k}^h(x,x')$ to make the dependence on the hash function h explicit. Using (3) we have

$$\begin{aligned}\overline{k}^h(x,x') &= \sum_j \sum_{i:h(i)=j} \phi_i(x) \sum_{i':h(i')=j} \phi'_i(x') \\ &= k(x,x') + \sum_{i,i':i\neq i'} \phi_i(x)\phi_{i'}(x')\delta_{h(i),h(i')} \end{aligned} \tag{5}$$

where δ is the Kronecker delta function. Taking the expectation with respect to the random choice of hash functions h we obtain the expected bias

$$\mathbf{E}_h[\overline{k}^h(x,x')] = \left(1-\tfrac{1}{n}\right)k(x,x') + \tfrac{1}{n}\sum_i \phi_i(x)\sum_{i'}\phi_{i'}(x')$$

Here we exploited the fact that for a random choice of hash functions the collision probability is $\frac{1}{n}$ uniformly over all pairs (i,j). Consequently $\overline{k}(x,x')$ is a biased estimator of the kernel matrix, with the bias decreasing inversely proportional to the number of hash bins.

The main change is a *rank-1* modification in the kernel matrix. Given the inherent high dimensionality of the estimation problem, a one dimensional change does not in general have a significant effect on generalization.

Straightforward (and tedious) calculation which is completely analogous to the above derivation leads to the following expression for the variance $\mathrm{Var}_h[\overline{k}^h(x,x')]$ of the hash kernel:

$$\mathrm{Var}_h[\overline{k}^h(x,x')] = \tfrac{n-1}{n^2}\left(k(x,x)k(x',x') + k^2(x,x') - 2\sum_i \phi_i^2(x)\phi_i^2(x')\right)$$

Key in the derivation is our assumption that the family of hash functions we are dealing with is pairwise independent.

As can be seen, the variance decreases $O(n^{-1})$ in the size of the values of the hash function. This means that we have an $O(n^{-\frac{1}{2}})$ convergence asymptotically to the expected value of the kernel.

4.2 Information Loss

One of the key fears of using hashing in machine learning is that hash collisions harm performance. For example, the well-known birthday paradox shows that if the hash function maps into a space of size n then with $O(n^{\frac{1}{2}})$ features a collision is likely. When a collision occurs, information is lost, which may reduce the achievable performance for a predictor.

Definition 1 *(Information Loss) A hash function h causes information loss on a distribution D with a loss function L if the expected minimum loss before hashing is less than the expected minimum loss after hashing:*

$$\min_f \underset{(x,y)\sim D}{E}[L(f(x),y))] < \min_g \underset{(x,y)\sim D}{E}[L(g(h(x)),y))]$$

Redundancy in features is very helpful in avoiding information loss. The redundancy can be explicit or systemic such as might be expected with a bag-of-words or substring representation. In the following we analyze explicit redundancy where a feature is mapped to two or more values in the space of size n. This can be implemented with a hash function by (for example) appending the string $i \in \{1,\ldots,c\}$ to feature f and then computing the hash of $f \circ i$ for the i-th duplicate.

The essential observation is that the information in a feature is only lost if all duplicates of the feature collide with other features. Given this observation, it's unsurprising that increasing the size of n by a constant multiple c and duplicating features c times makes collisions with all features unlikely. It's perhaps more surprising that when keeping the size of n *constant* and duplicating features, the probability of information loss can go *down*.

Theorem 2 *For a random function mapping l features duplicated c times into a space of size n, for all loss functions L and and distributions D on n features, the probability (over the random function) of no information loss is at least:*

$$1 - l[1 - (1 - c/n)^c + (lc/n)^c].$$

To see the implications consider $l = 10^5$ and $n = 10^8$. Without duplication, a birthday paradox collision is virtually certain. However, if $c = 2$, the probability of information loss is bounded by about 0.404, and for $c = 3$ it drops to about 0.0117.

Proof The proof is essentially a counting argument with consideration of the fact that we are dealing with a hash *function* rather than a random variable. It is structurally similar to the proof for a Bloom filter (Bloom, 1970), because the essential question we address is: "What is a lower bound on the probability that all features have one duplicate not colliding with any other feature?"

Fix a feature f. We'll argue about the probability that all c duplicates of f collide with other features.

For feature duplicate i, let $h_i = h(f \circ i)$. The probability that $h_i = h(f' \circ i')$ for some other feature $f' \circ i'$ is bounded by $(l-1)c/n$ because the probability for each other mapping of a collision is $1/n$ by the assumption that h is a random function, and the union bound applied to the $(l-1)c$ mappings of other features yields $(l-1)c/n$. Note that we do not care about a collision of two duplicates of the same feature, because the feature value is preserved.

The probability that all duplicates $1 \le i \le c$ collide with another feature is bounded by $(lc/n)^c + 1 - (1 - c/n)^c$. To see this, let $c' \le c$ be the number of distinct duplicates of f after collisions. The probability of a collision with the first of these is bounded by $\frac{(l-1)c}{n}$. Conditioned on this collision, the probability of the next collision is at most $\frac{(l-1)c-1}{n-1}$, where 1 is subtracted because the first location is fixed. Similarly, for the ith duplicate, the probability is $\frac{(l-1)c-(i-1)}{n-(i-1)}$. We can upper bound each term as $\frac{lc}{n}$, implying the probability of all c' duplicates colliding with other features is at most $(lc/n)^{c'}$. The probability that $c' = c$ is the probability that none of the duplicates of f collide, which is $\frac{(n-1)!}{n^c(n-c-1)!} \ge ((n-c)/n)^c$. If we pessimistically assume that $c' < c$ implies that every duplicate collides with another feature, then

$$\begin{aligned} \mathrm{P}(\mathrm{coll}) &\le \mathrm{P}(\mathrm{coll}|c' = c)\,\mathrm{P}(c' = c) + \mathrm{P}(c' \ne c) \\ &\le (lc/n)^c + 1 - ((l-c)/l)^c. \end{aligned}$$

Simplification gives $(lc/n)^c + 1 - (1 - c/n)^c$ as claimed. Taking a union bound over all l features, we get that the probability any feature has all duplicates collide is bounded by $l[1 - (1 - c/n)^c +$

$(lc/n)^c]$. ■

4.3 Rate of Convergence

As a first step note that any convergence bound only depends *logarithmically* on the size of the kernel matrix.

Theorem 3 *Assume that the probability of deviation between the hash kernel and its expected value is bounded by an exponential inequality via*

$$\mathrm{P}\left[\left|\overline{k}^h(x,x') - \mathbf{E}_h\left[\overline{k}^h(x,x')\right]\right| > \varepsilon\right] \leq c\exp(-c'\varepsilon^2 n)$$

for some constants c, c' depending on the size of the hash and the kernel used. In this case the error ε arising from ensuring the above inequality, with probability at least $1-\delta$, for m observations and M classes (for a joint feature map $\phi(x,y)$, is bounded by

$$\varepsilon \leq \sqrt{(2\log(m+1) + 2\log(M+1) - \log\delta + \log c - 2\log 2)/nc'}.$$

Proof Apply the union bound to the kernel matrix of size $(mM)^2$, that is, to all $T := m(m+1)M(M+1)/4$ unique elements. Solving

$$Tc\exp(-c'\varepsilon^2 n) = \delta,$$

we get the bound on ε is

$$\sqrt{\frac{\log(Tc) - \log\delta}{c'n}}. \tag{6}$$

Bounding $\log(Tc)$ from above

$$\log(Tc) = \log T + \log c \leq 2\log(m+1) + 2\log(M+1) + \log c - 2\log 2,$$

and plugging it into (6) yields the result. ■

4.4 Generalization Bound

The hash kernel approximates the original kernel by big storage and computation saving. An interesting question is whether the generalization bound on the hash kernel will be **much worse** than the bound obtained on the original kernel.

Theorem 4 (Generalization Bound) *For binary class SVM, let $\overline{K} = \left\langle \overline{\phi}(x), \overline{\phi}(x') \right\rangle, K = \langle \phi(x), \phi(x') \rangle$ be the hash kernel matrix and original kernel matrix. Assume there exists $b \geq 0$ such that $b \geq tr(\overline{K}) - tr(K)$. Let $\mathcal{F}$ be the class of functions mapping from $\mathcal{X} \times \mathcal{Y}$ to $\mathbb{R}$ given by $f(x,y) = -yg(x)$, where g is a linear function in a kernel-defined feature space with norm at most 1. For any size m sample $\{(x_1,y_1),\ldots,(x_m,y_m)\}$ drawn i.i.d. from data distribution D over $\mathcal{X} \times \mathcal{Y}$ and for any*

$\delta \in (0,1)$ *and any* $\gamma > 0$*, with probability at least* $1-\delta$*, we have*

$$P(y \neq sgn(g(x))) \leq \frac{1}{m\gamma}\sum_{i=1}^{m}\xi_i + \frac{4}{m\gamma}\sqrt{tr(\overline{K})} + 3\sqrt{\frac{ln(\frac{2}{\delta})}{2m}} \tag{7}$$

$$\leq \frac{1}{m\gamma}\sum_{i=1}^{m}\xi_i + \frac{4}{m\gamma}\sqrt{tr(K)} + \frac{4}{m\gamma}\sqrt{b} + 3\sqrt{\frac{ln(\frac{2}{\delta})}{2m}}, \tag{8}$$

where $\xi_i = \max\{0, \gamma - y_i g(x_i)\}$*, and* $g(x_i) = \left\langle w, \overline{\phi}(x_i, y_i)\right\rangle$.

Proof The standard Rademacher bound states that for any size m sample drawn i.i.d from D, for any $\delta \in (0,1)$ and any $\gamma > 0$, with probability at least $1-\delta$, the true error of the binary SVM ,whose kernel matrix is denoted as K', can be bounded as follows:

$$P(y \neq sgn(g(x))) \leq \frac{1}{m\gamma}\sum_{i=1}^{m}\xi_i + \frac{4}{m\gamma}\sqrt{tr(K')} + 3\sqrt{\frac{ln(\frac{2}{\delta})}{2m}}. \tag{9}$$

We refer the reader to Theorem 4.17 in Shawe-Taylor and Cristianini (2004) for a detailed proof. The inequality (7) follows by letting $K' = \overline{K}$. Because $tr(\overline{K}) \leq b + tr(K) \leq (\sqrt{b} + \sqrt{tr(K)})^2$, we have $\frac{4}{m\gamma}\sqrt{tr(\overline{K})} \leq \frac{4}{m\gamma}\sqrt{tr(K)} + \frac{4}{m\gamma}\sqrt{b}$. Plugging above inequality into inequality (7) gives inequality (8). So the theorem holds. ■

The approximation quality depends on the both the feature and the hash collision. From the definition of hash kernel (see (3)), the feature entries with the same hash value will add up and map to a new feature entry indexed by the hash value. The higher collision of the hash has, the more entries will be added up. If the feature is sparse, the added up entries are mostly zeros. So the difference of the maximum and the sum of the entries is reasonably small. In this case, the hash kernel gives a good approximation of the original kernel, so b is reasonably small. Thus the generalization error does not increase much as the collision increases. This is verified in the experiment section in Table 4—increasing the collision rate from 0.82% to 94.31% only slightly worsens the test error (from 5.586% to 6.096%).

Moreover, Theorem 4 shows us the generalization bounds on the hash kernel and the original kernel only differ by $O((m\gamma)^{-1})$. This means that when our data set is large, the difference can be ignored. A surprising result as we shall see immediately is that in fact, the difference on the generalization bound is always nearly zero regardless of m, γ.

4.5 The Scaling Factor Effect

An interesting observation on Theorem 4 is that, if we use a new feature mapping $\phi' = a\phi$, where $a \in [0,1)$, it will make the bias term in (5) small. As we decrease a enough, the bias term can be arbitrarily small. Indeed, it vanishes when $a = 0$. It seems that thus we can get a much tighter bound according to Theorem 4—the term with b vanishes when $a = 0$. So is there an optimal scaling factor a that maximizes the performance? It turns out that any nonzero a doesn't effect the performance at all, although it does tighten the bound.

Let's take a closer look at the hash kernel binary SVM, which can be formalized as

$$\min_{w} \quad \frac{\lambda||w||^2}{2} + \sum_{i=1}^{m}\max\{0, 1 - y_i\left\langle w, \overline{\phi}(x_i, y_i)\right\rangle\}. \tag{10}$$

Applying a new feature mapping $\hat{\phi} = a\bar{\phi}$ gives

$$\min_{\hat{w}} \quad \frac{\hat{\lambda}||\hat{w}||^2}{2} + \sum_{i=1}^{m} \max\{0, 1 - y_i \left\langle \hat{w}, \bar{\phi}(x_i, y_i) \right\rangle\}, \tag{11}$$

where $\hat{\lambda} = \frac{\lambda}{a^2}$ and $\hat{w} = aw$. $\hat{\lambda}$ is usually determined by model selection. As we can see, given a training data set, the solutions to (10) and (11) are exactly identical for $a \neq 0$. When $a = 0$, $g(x) \equiv 0$ for all x, w, and the prediction degenerates to random guessing. Moreover, the generalization bound can be tighten by applying a small a nearly zero. This shows that hashing kernel SVM has nearly the same generalization bound as the original SVM in theory.

5. Graphlet Kernels

Denote by G a graph with vertices $V(G)$ and edges $E(G)$. Several methods have been proposed to perform classification on such graphs. Most recently, Przulj (2007) proposed to use the distribution over graphlets, that is, subgraphs, as a characteristic property of the graph. Unfortunately, brute force evaluation does not allow calculation of the statistics for graphlets of size more than 5, since the cost for exact computation scales exponentially in the graphlet size.

In the following we show that sampling and hashing can be used to make the analysis of larger subgraphs tractable in practice. For this denote by $S \subseteq G$ an induced subgraph of G, obtained by restricting ourselves to only $V(S) \subseteq V(G)$ vertices of G and let $\#_S(G)$ be the number of times S occurs in G. This suggests that the feature map $G \to \phi(G)$, where $\phi_S(G) = \#_S(G)$ will induce a useful kernel: adding or removing an edge (i, j) only changes the properties of the subgraphs using the pair (i, j) as part of their vertices.

5.1 Counting and Sampling

Depending on the application, the distribution over the counts of subgraphs may be significantly skewed. For instance, in sparse graphs we expect the fully disconnected subgraphs to be considerably overrepresented. Likewise, whenever we are dealing with almost complete graphs, the distribution may be skewed towards the other end (i.e., most subgraphs will be complete). To deal with this, we impose weights $\beta(k)$ on subgraphs containing k edges $|E(S)|$.

To deal with the computational complexity issue simultaneously with the issue of reweighting the graphs S we simply replace explicit counting with sampling from the distribution

$$\mathrm{P}(S|G) = c(G)\beta(|E(S)|) \tag{12}$$

where $c(G)$ is a normalization constant. Samples from $\mathrm{P}(S|G)$ can be obtained by a Markov-Chain Monte Carlo approach.

Lemma 5 *The following MCMC sampling procedure has the stationary distribution (12).*

1. *Choose a random vertex, say i, of S uniformly.*
2. *Add a vertex j from $G \backslash S_i$ to S_i with probability $c(S_i, G)\beta(|E(S_{ij})|)$.*

Here S_i denotes the subgraph obtained by removing vertex i from S, and S_{ij} is the result of adding vertex j to S_i.

Note that sampling over j is easy: all vertices of G which do not share an edge with $S\backslash i$ occur with the same probability. All others depend only on the number of joining edges. This allows for easy computation of the normalization $c(S_i, G)$.

Proof We may encode the sampling rule via

$$T(S_{ij}|S,G) = \frac{1}{k} c(S_i, G)\beta(|E(S_{ij})|)$$

where $c(S_i, G)$ is a suitable normalization constant. Next we show that T satisfies the balance equations and therefore can be used as a proposal distribution with acceptance probability 1.

$$\begin{aligned}&\frac{T(S_{ij}|S,G)\,\mathrm{P}(S)}{T(S|S_{ij},G)\,\mathrm{P}(S_{ij})}\\ =&\frac{k^{-1}c(S_i,G)\beta(|E(S_{ij})|)c(G)\beta(|E(S)|)}{k^{-1}c(S_{ij,j},G)\beta(|E(S_{ij,ji})|)c(G)\beta(|E(S_{ij})|)} = 1.\end{aligned}$$

This follows since $S_{ij,j} = S_i$ and likewise $S_{ij,ji} = S$. That is, adding and removing the same vertex leaves a graph unchanged. ■

In summary, we obtain an algorithm which will readily draw samples S from $\mathrm{P}(S|G)$ to characterize G.

5.2 Dependent Random Variables

The problem with sampling from a MCMC procedure is that the random variables are *dependent* on each other. This means that we cannot simply appeal to Chernoff bounds when it comes to averaging. Before discussing hashing we briefly discuss averages of dependent random variables:

Definition 6 (Bernoulli Mixing) *Denote by Z a stochastic process indexed by $t \in \mathbb{Z}$ with probability measure* P *and let Σ_n be the σ-algebra on Z_t with $t \in \mathbb{Z}\backslash 1, \ldots, n-1$. Moreover, denote by* P_- *and* P_+ *the probability measures on the negative and positive indices t respectively. The mixing coefficient β is*

$$\beta(n, \mathrm{P}_X) := \sup_{A \in \Sigma_n} \left| \mathrm{P}(A) - \mathrm{P}_- \times \mathrm{P}_+(A) \right|.$$

If $\lim_{n\to\infty} \beta(n, \mathrm{P}_z) = 0$ we call Z to be β-mixing.

That is, $\beta(n, \mathrm{P}_X)$ measures how much dependence a sequence has when cutting out a segment of length n. Nobel and Dembo (1993) show how such mixing processes can be related to iid observations.

Theorem 7 *Assume that* P *is β-mixing. Denote by* P^* *the product measure obtained from $\ldots \mathrm{P}_t \times \mathrm{P}_{t+1} \ldots$ Moreover, denote by $\Sigma_{l,n}$ the σ-algebra on $Z_n, Z_{2n}, \ldots, Z_{ln}$. Then the following holds:*

$$\sup_{A \in \Sigma_{l,n}} |\mathrm{P}(A) - \mathrm{P}^*(A)| \leq l\beta(n, \mathrm{P}).$$

This allows us to obtain bounds for expectations of variables drawn from P rather than P^*.

Theorem 8 *Let* P *be a distribution over a domain* $\mathcal{X}$ *and denote by* $\phi : \mathcal{X} \to \mathcal{H}$ *a feature map into a Hilbert Space with* $\langle \phi(x), \phi(x') \rangle \in [0,1]$. *Moreover, assume that there is a* β*-mixing MCMC sampler of* P *with distribution* P^{MC} *from which we draw* l *observations* x_{in} *with an interleave of* n *rather than sampling from* P *directly. Averages with respect to* P^{MC} *satisfy the following with probability at least* $1-\delta$*:*

$$\left\| \mathop{\mathbf{E}}_{x \sim \mathrm{P}(x)}[\phi(x)] - \frac{1}{l}\sum_{i=1}^{l} \phi(x_{in}) \right\| \leq l\beta(n, \mathrm{P}^{\mathrm{MC}}) + \frac{2+\sqrt{\log \frac{2}{\delta}}}{\sqrt{l}}.$$

Proof Theorem 7, the bound on $\|\phi(x)\|$, and the triangle inequality imply that the expectations with respect to P^{MC} and P^* only differ by $l\beta$. This establishes the first term of the bound. The second term is given by a uniform convergence result in Hilbert Spaces from Altun and Smola (2006). ■

Hence, sampling from a MCMC sampler for the purpose of approximating inner products is sound, provided that we only take sufficiently independent samples (i.e., a large enough n) into account. The translation of Theorem 8 into bounds on inner products is straightforward, since

$$\begin{aligned} &|\langle x, y \rangle - \langle x', y' \rangle| \\ \leq &\left\|x - x'\right\| \|y\| + \left\|y - y'\right\| \|x\| + \left\|x - x'\right\| \left\|y - y'\right\|. \end{aligned}$$

5.3 Hashing and Subgraph Isomorphism

Sampling from the distribution over subgraphs $S \in G$ has two serious problems in practice which we will address in the following: firstly, there are several graphs which are isomorphic to each other. This needs to be addressed with a graph isomorphism tester, such as Nauty (McKay, 1984). For graphs up to size 12 this is a very effective method. Nauty works by constructing a lookup table to match isomorphic objects.

However, even after the graph isomorphism mapping we are still left with a sizable number of distinct objects. This is where a hash map on data streams comes in handy. It obviates the need to store any intermediate results, such as the graphs S or their unique representations obtained from Nauty. Finally, we combine the convergence bounds from Theorem 8 with the guarantees available for hash kernels to obtain the approximate graph kernel.

Note that the two randomizations have very different purposes: the sampling over graphlets is done as a way to approximate the *extraction* of features whereas the compression via hashing is carried out to ensure that the representation is computationally efficient.

6. Experiments

To test the efficacy of our approach we applied hashing to the following problems: first we used it for classification on the Reuters RCV1 data set as it has a relatively large feature dimensionality. Secondly, we applied it to the DMOZ ontology of topics of webpages[1] where the number of topics is high. The third experiment—Biochemistry and Bioinformatics Graph Classification uses our hashing scheme, which makes comparing all possible subgraph pairs tractable, to compare graphs (Vishwanathan et al., 2007a). On publicly available data sets like MUTAG and PTC as well as on

1. Dmoz L2 denotes non-parent topic data in the top 2 levels of the topic tree and Dmoz L3 denotes non-parent topic data in the top 3 levels of the topic tree.

Data Sets	#Train	#Test	#Labels
RCV1	781,265	23,149	2
Dmoz L2	4,466,703	138,146	575
Dmoz L3	4,460,273	137,924	7,100

Table 1: Text data sets. #X denotes the number of observations in X.

Algorithm	Pre	TrainTest	Error %
BSGD	303.60s	10.38s	6.02
VW	303.60s	87.63s	5.39
VWC	303.60s	5.15s	5.39
HK	0s	25.16s	5.60

Table 2: Runtime and Error on RCV1. BSGD: Bottou's SGD. VW: Vowpal Wabbit without cache. VWC: Vowpal Wabbit using cache file. HK: hash kernel with feature dimension 2^{20}. Pre: preprocessing time. TrainTest: time to load data, train and test the model. Error: misclassification rate. Apart from the efficacy of hashing operation itself, the gain of speed is also due to a multi-core implementation—hash kernel uses 4-cores to access the disc for online hash feature generation. For learning and testing evaluation, all algorithms use single-core.

the biologically inspired data set DD used by Vishwanathan et al. (2007a), our method achieves the best known accuracy.

In both RCV1 and Dmoz, we use linear kernel SVM with stochastic gradient descent (SGD) as the workhorse. We apply our hash kernels and random projection (Achlioptas, 2003) to the SGD linear SVM. We don't apply the approach in Rahimi and Recht (2008) since it requires a shift-invariant kernel $k(x,y) = k(x-y)$, such as RBF kernel, which is not applicable in this case. In the third experiment, existing randomization approaches are not applicable since enumerating all possible subgraphs is intractable. Instead we compare hash kernel with existing graph kernels: random walk kernel, shortest path kernel and graphlet kernel (see Borgwardt et al. 2007).

6.1 Reuters Articles Categorization

We use the Reuters RCV1 binary classification data set (Lewis et al., 2004). 781,265 articles are used for training by stochastic gradient descent (SGD) and 23,149 articles are used for testing. Conventionally one would build a bag of words representation first and calculate exact term frequency / inverse document frequency (TF-IDF) counts from the contents of each article as features. The problem is that the TF calculation needs to maintain a very large dictionary throughout the whole process. Moreover, it is impossible to extract features online since the entire vocabulary dictionary is usually unobserved during training. Another disadvantage is that calculating exact IDF requires us to preprocess all articles in a first pass. This is not possible as articles such as news may vary daily.

However, it suffices to compute TF and IDF approximately as follows: using hash features, we no longer require building the bag of words. Every word produces a hash key which is the dimension index of the word. The frequency is recorded in the dimension index of its hash key. Therefore,

Algorithm	Dim	Pre	TrainTest	orgTrainSize	newTrainSize	Error %
RP	2^8	748.30s	210.23s	423.29Mb	1393.65Mb	29.35%
	2^9	1079.30s	393.46s	423.29Mb	2862.90Mb	25.08%
	2^{10}	1717.30s	860.95s	423.29Mb	5858.48Mb	19.86%
HK	2^8	0s	22.82s	NA	NA	17.00%
	2^9	0s	24.19s	NA	NA	12.32%
	2^{10}	0s	24.41s	NA	NA	9.93%

Table 3: Hash kernel vs. random projections with various feature dimensionalities on RCV1. RP: random projections in Achlioptas (2003). HK: hash kernel. Dim: dimension of the new features. Pre: preprocessing time. TrainTest: time to load data, train and test the model. orgTrainSize: compressed original training feature file size. newTrainSize: compressed new training feature file size. Error: misclassification rate. NA: not applicable. In hash kernel there is no preprocess step, so there is no original/new feature files. Features for hash kernel are built up online via accessing the string on disc. The disc access time is taken into account in **TrainTest**. Note that the TrainTest for random projection time increases as the new feature dimension increases, whereas for hash kernel the TrainTest is almost independent of the feature dimensionality.

Dim	#Unique	Collision %	Error %
2^{24}	285614	0.82	5.586
2^{22}	278238	3.38	5.655
2^{20}	251910	12.52	5.594
2^{18}	174776	39.31	5.655
2^{16}	64758	77.51	5.763
2^{14}	16383	94.31	6.096

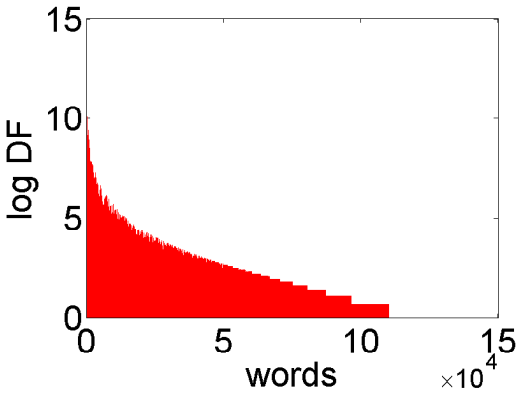
Table 4: Influence of new dimension on Reuters (RCV1) on collision rates (reported for both training and test set combined) and error rates. Note that there is no noticeable performance degradation even for a 40% collision rate.

every article has a frequency count vector as TF. This TF is a much denser vector which requires no knowledge of the vocabulary. IDF can be approximated by scanning a smaller *part* of the training set.

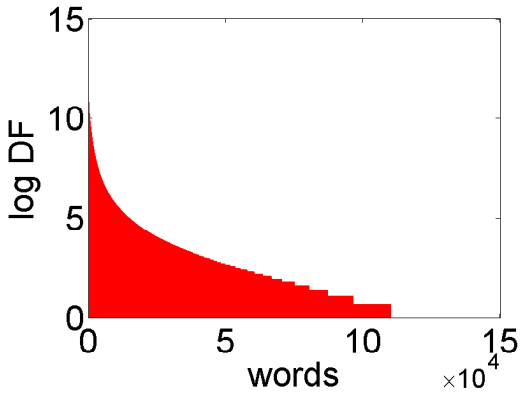
A quantile-quantile plot in Figure 1 shows that this approximation is justified—the dependency between the statistics on the subset (200k articles) and the full training set (800k articles) is perfectly linear.

We compare the hash kernel with Leon Bottou's Stochastic Gradient Descent SVM[2] (BSGD), Vowpal Wabbit (Langford et al., 2007) (VW) and Random Projections (RP) (Achlioptas, 2003). Our hash scheme is generating features online. BSGD is generating features offline and learning them online. VW uses BSGD's preprocessed features and creates further features online. Caching speeds up VW considerably. However, it requires one run of the original VW code for this purpose. RP uses BSGD's preprocessed features and then creates the new projected lower dimension features. Then it uses BSGD for learning and testing. We compare these algorithms on RCV1 in Table 2.

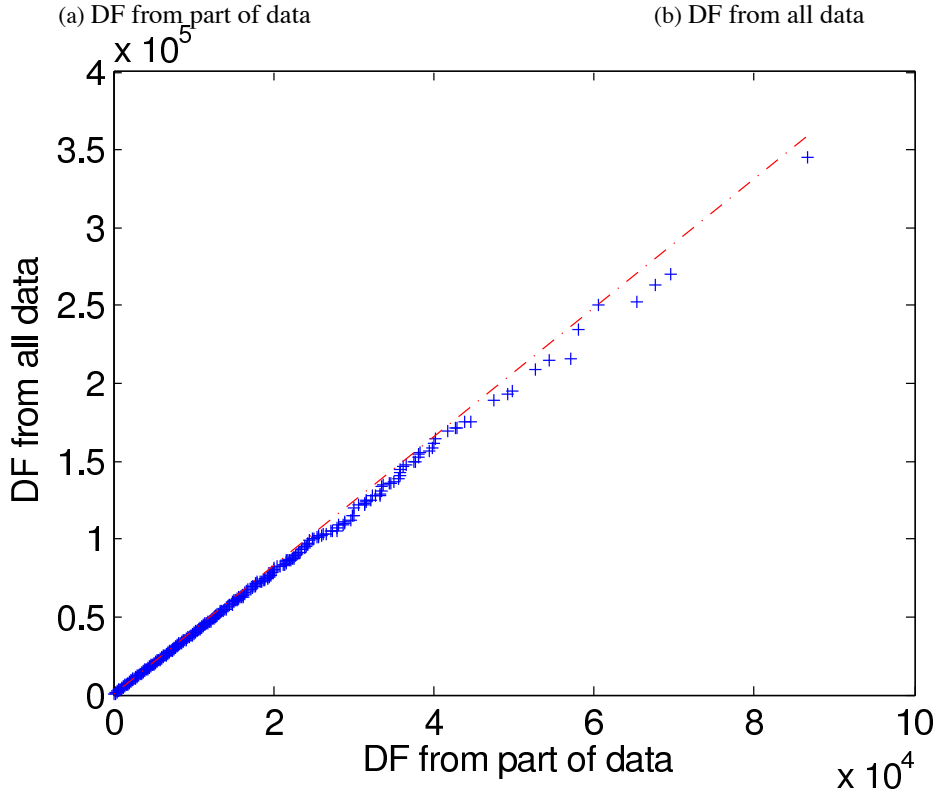
2. Code can be found at `http://leon.bottou.org/projects/sgd`.



(a) DF from part of data



(b) DF from all data



(c) Quantile-Quantile plot

Figure 1: Quantile-quantile plot of the DF counts computed on a subset (200k documents) and the full data set (800k documents). DF(t) is the number of documents in a collection containing word t.

Table 2. RP is not included in this table because it would be intractable to run it with the same feature dimensionality as HK for a fair comparison. As can be seen, the preprocessing time of BSGD and VW is considerably longer compared to the time for training and testing, due to the TF-IDF calculation which is carried out offline. For a fair comparison, we measure the time for feature loading, training and testing together. It can also be seen that the speed of online feature generation

	HLF (2^{28})		**HLF** (2^{24})		**HF**		**no hash**	**U base**	**P base**
	error	mem	error	mem	error	mem	mem	error	error
L2	30.12	2G	30.71	0.125G	31.28	2.25G (2^{19})	7.85G	99.83	85.05
L3	52.10	2G	53.36	0.125G	51.47	1.73G (2^{15})	96.95G	99.99	86.83

Table 5: Misclassification and memory footprint of hashing and baseline methods on DMOZ. HLF: joint hashing of labels and features. HF: hash features only. no hash: direct model (not implemented as too large, hence only memory estimates—we have 1,832,704 unique words). U base: baseline of uniform classifier. P base: baseline of majority vote. mem: memory used for the model. Note: the memory footprint in HLF is essentially independent of the number of classes used.

	HLF		**KNN**			**Kmeans**		
	2^{28}	2^{24}	S= 3%	6%	9%	S= 3%	6%	9%
L2	69.88	69.29	50.23	52.59	53.81	42.29	42.96	42.76
L3	47.90	46.64	30.93	32.67	33.71	31.63	31.56	31.53

Table 6: Accuracy comparison of hashing, KNN and Kmeans. HLF: joint hashing of labels and features. KNN: apply K Nearest Neighbor on sampled training set as search set. Kmeans: apply Kmeans on sampled training set to do clustering and then take its majority class as predicted class. S is the sample size which is the percentage of the entire training set.

is considerable compared to disk access. Table 2 shows that the test errors for hash kernel, BSGD and VW are competitive.

In table 3 we compare hash kernel to RP with different feature dimensions. As we can see, the error reduces as the new feature dimension increases. However, the error of hash kernel is always much smaller (by about 10%) than RP given the same new dimension. An interesting thing is that the new feature file created after applying RP is much bigger than the original one. This is because the projection maps the original sparse feature to a dense feature. For example, when the feature dimension is 2^{10}, the compressed new feature file size is already 5.8G. Hash kernel is much more efficient than RP in terms of speed, since to compute a hash feature one requires only $O(d_{nz})$ hashing operations, where d_{nz} is the number of non-zero entries. To compute a RP feature one requires $O(dn)$ operations, where d is the original feature dimension and and n is the new feature dimension. And with RP the new feature is always dense even when n is big, which further increases the learning and testing runtime. When $d_{nz} \ll d$ such as in text process, the difference is significant. This is verified in our experiment (see in Table 3). For example, hash kernel (including **Pre** and **TrainTest**) with 2^{10} feature size is over 100 times faster than RP.

Furthermore, we investigate the influence of the new feature dimension on the misclassification rate. As can be seen in Table 4, when the feature dimension decreases, the collision and the error rate increase. In particular, a 2^{24} dimension causes almost no collisions. Nonetheless, a 2^{18} dimension which has almost 40% collisions performs equally well on the problem. This leads to rather memory-efficient implementations.

Data	Algorithm	Dim	Pre	TrainTest	Error %
L2	RP	2^7	779.98s	1258.12s	82.06%
	RP	2^8	1496.22s	3121.66s	72.66%
	RP	2^9	2914.85s	8734.25s	62.75%
	HK	2^7	0s	165.13s	62.28%
	HK	2^8	0s	165.63s	55.96%
	HK	2^9	0s	174.83s	50.98%
L3	RP	2^7	794.23s	18054.93s	89.46%
	RP	2^8	1483.71s	38613.51s	84.06%
	RP	2^9	2887.55s	163734.13s	77.25%
	HK	2^7	0s	1573.46s	76.31%
	HK	2^8	0s	1726.67s	71.93%
	HK	2^9	0s	1812.98s	67.18%

Table 7: Hash kernel vs. random projections with various feature dimensionalities on Dmoz. RP: random projections in Achlioptas (2003). HK: hash kernel. Dim: dimension of the new features. Pre: preprocessing time—generation of the random projected features. TrainTest: time to load data, train and test the model. Error: misclassification rate. Note that the TrainTest time for random projections increases as the new feature dimension increases, whereas for hash kernel the TrainTest is almost independent of the feature dimensionality. Moving the dimension from 2^8 to 2^9 the increasing in processing time of RP is not linear—we suspect this is because with 2^8 the RP model has $256 \times 7100 \times 8 \approx 14$MB, which is small enough to fit in the CPU cache (we are using a 4-cores cpu with a total cache size of 16MB), while with 2^9 the model has nearly 28MB, no longer fitting in the cache.

6.2 Dmoz Websites Multiclass Classification

In a second experiment we perform topic categorization using the DMOZ topic ontology. The task is to recognize the topic of websites given the short descriptions provided on the webpages. To simplify things we categorize only the leaf nodes (Top two levels: L2 or Top three levels: L3) as a flat classifier (the hierarchy could be easily taken into account by adding hashed features for each part of the path in the tree). This leaves us with 575 leaf topics on L2 and with 7100 leaf topics on L3.

Conventionally, assuming M classes and l features, training M different parameter vectors w requires $O(Ml)$ storage. This is infeasible for massively multiclass applications. However, by hashing data and labels jointly we are able to obtain an efficient joint representation which makes the implementation computationally possible.

As can be seen in Table 5 joint hashing of features and labels is very attractive in items of memory usage and in many cases is necessary to make large multiclass categorization computationally feasible at all (naive online SVM ran out of memory). In particular, hashing features only produces worse results than joint hashing of labels and features. This is likely due to the increased collision rate: we need to use a smaller feature dimension to store the class dependent weight vectors explicitly.

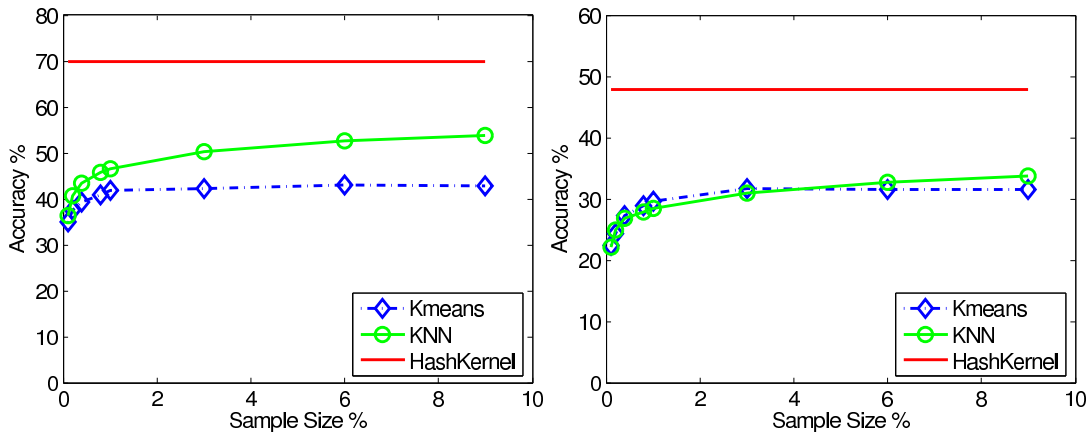


Figure 2: Test accuracy comparison of KNN and Kmeans on Dmoz with various sample sizes. Left: results on L2. Right: results on L3. Hash kernel (2^{28}) result is used as an upper bound.

Next we compare hash kernel with K Nearest Neighbor (KNN) and Kmeans. Running the naive KNN on the entire training set is very slow.[3] Hence we introduce sampling to KNN. We first sample a subset from the entire training set as search set and then do KNN classification. To match the scheme of KNN, we use sampling in Kmeans too. Again we sample from the entire training set to do clustering. The number of clusters is the minimal number of classes which have at least 90% of the documents. Each test example is assigned to one of the clusters, and we take the majority class of the cluster as the predicted label of the test example. The accuracy plot in Figure 2 shows that in both Dmoz L2 and L3, KNN and Kmeans with various sample sizes get test accuracies of 30% to 20% off than the upper bound accuracy achieved by hash kernel. The trend of the KNN and Kmeans accuracy curve suggests that the bigger the sample size is, the less accuracy increment can be achieved by increasing it. A numerical result with selected sample sizes is reported in Table 6.

We also compare hash kernel with RP with various feature dimensionality on Dmoz. Here RP generates the random projected feature first and then does online learning and testing. It uses the same 4-cores implementation as hash kernel does. The numerical result with selected dimensionalities is in Table 7. It can be seen that hash kernel is not only much faster but also has much smaller error than RP given the same feature dimension. Note that both hash kernel and RP reduce the error as they increase the feature dimension. However, RP can't achieve competitive error to what hash kernel has in Table 5, simply because with large feature dimension RP is too slow—the estimated run time for RP with dimension 2^{19} on dmoz L3 is 2000 days.

Furthermore we investigate whether such a good misclassification rate is obtained by predicting well only on a few dominant topics. We reorder the topic histogram in accordance to ascending error rate. Figure 3 shows that hash kernel does very well on the first one hundred topics. They correspond to easy categories such as language related sets "World/Italiano","World/Japanese","World/Deutsch".

3. In fact the complexity of KNN is $O(N \times T)$, where N, T are the size of the training set and the testing set. We estimate the running time for the original KNN, in a batch processing manner ignoring the data loading time, is roughly 44 days on a PC with a 3.2GHz cpu.

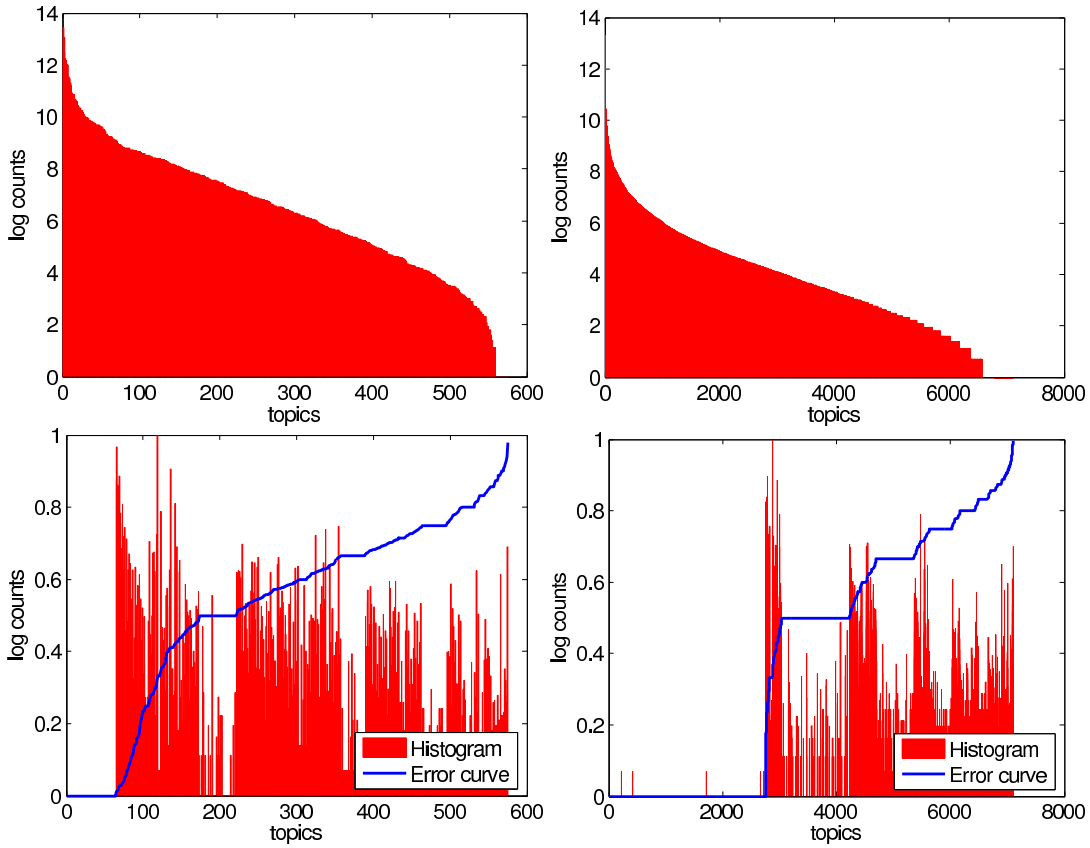


Figure 3: Left: results on L2. Right: results on L3. Top: frequency counts for topics as reported on the training set (the test set distribution is virtually identical). We see an exponential decay in counts. Bottom: log-counts and error probabilities on the test set. Note that the error is reasonably evenly distributed among the size of the classes (besides a number of near empty classes which are learned perfectly).

6.3 Biochemistry and Bioinformatics Graph Classification

For the final experiment we work with graphs. The benchmark data sets we used here contain three real-world data sets: two molecular compounds data sets, Debnath et al. (1991) and PTC (Toivonen et al., 2003), and a data set for protein function prediction task (DD) from Dobson and Doig (2003). In this work we used the unlabeled version of these graphs, see, for example, Borgwardt et al. (2007).

All these data sets are made of sparse graphs. To capture the structure of the graphs, we sampled connected subgraphs with varying number of nodes, from $n = 4$ to $n = 9$. We used graph isomorphism techniques, implemented in Nauty (McKay, 1984) for getting a canonically-labeled isomorph of each sampled subgraph. The feature vector of each example (graph) is composed of the number of times each canonical isomorph was sampled. Each graph was sampled 10000 times for each of $n = 4, 5 \ldots 9$. Note that the number of connected unlabeled graphs grows exponentially fast with the number of nodes, so the sampling is extremely sparse for large values of n. For this reason we normalized the counts so that for each data set each feature of $\phi(x)$ satisfies $1 \geq \phi(x) \geq 0$.

Data Sets	RW	SP	GKS	GK	HK	HKF
MUTAG	0.719	0.813	0.819	0.822	0.855	0.865
PTC	0.554	0.554	0.594	0.597	0.606	0.635
DD	>24h	>24h	0.745	>24h	0.799	0.841

Table 8: Classification accuracy on graph benchmark data sets. RW: random walk kernel, SP: shortest path kernel, GKS = graphlet kernel sampling 8497 graphlets, GK: graphlet kernel enumerating all graphlets exhaustively, HK: hash kernel, HKF: hash kernel with feature selection. '>24h' means computation did not finish within 24 hours.

Feature	All		Selection	
STATS	ACC	AUC	ACC	AUC
MUTAG	0.855	0.93	0.865	0.912
PTC	0.606	0.627	0.635	0.670
DD	0.799	0.81	0.841	0.918

Table 9: Non feature selection vs feature selection for hash kernel. All: all features. Selection: feature selection; ACC: accuracy; AUC: Area under ROC.

We compare the proposed hash kernel (with/without feature selection) with random walk kernel, shortest path kernel and graphlet kernel on the benchmark data sets. From Table 8 we can see that the hash Kernel even without feature selection still significantly outperforms the other three kernels in terms of classification accuracy over all three benchmark data sets.

The dimensionality of the canonical isomorph representation is quite high and many features are extremely sparse, a feature selection step was taken that removed features suspected as non-informative. To this end, each feature was scored by the absolute vale of its correlation with the target. Only features with scores above median were retained. As can be seen in Table 9 feature selection on hash kernel can furthermore improve the test accuracy and area under ROC.

7. Discussion

In this paper we showed that hashing is a computationally attractive technique which allows one to approximate kernels for very high dimensional settings efficiently by means of a sparse projection into a lower dimensional space. In particular for multiclass categorization this makes all the difference in terms of being able to implement problems with thousands of classes in practice on large amounts of data and features.

References

D. Achlioptas. Database-friendly random projections:johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66:671C687, June 2003.

Y. Altun and A.J. Smola. Unifying divergence minimization and statistical inference via convex duality. In H.U. Simon and G. Lugosi, editors, *Proc. Annual Conf. Computational Learning Theory*, LNCS, pages 139–153. Springer, 2006.

C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer, New York, 1984.

B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422C426, July 1970.

K. M. Borgwardt, H.-P. Kriegel, S. V. N. Vishwanathan, and N. Schraudolph. Graph kernels for disease outcome prediction from protein-protein interaction networks. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Tiffany Murray, and Teri E Klein, editors, *Proceedings of the Pacific Symposium of Biocomputing 2007*, Maui Hawaii, January 2007. World Scientific.

C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381, Cambridge, MA, 1997. MIT Press.

G. Cormode and M. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *LATIN: Latin American Symposium on Theoretical Informatics*, 2004.

A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *J Med Chem*, 34:786–797, 1991.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: A kernel-based perceptron on a fixed budget. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

P. D. Dobson and A. J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *J Mol Biol*, 330(4):771–783, Jul 2003.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *JMLR*, 2:243–264, Dec 2001.

K. Ganchev and M. Dredze. Small statistical models by random feature mixing. In *workshop on Mobile NLP at ACL*, 2008.

P. Indyk and R. Motawani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the* 30th *Symposium on Theory of Computing*, pages 604–613, 1998.

L. Kontorovich. A universal kernel for learning regular languages. In *Machine Learning in Graphs*, 2007.

J. Langford, L. Li, and A. Strehl. Vowpal wabbit online learning project (technical report). Technical report, 2007.

D.D. Lewis, Y. Yang, T.G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

P. Li, K.W. Church, and T.J. Hastie. Conditional random sampling: A sketch-based sampling technique for sparse data. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 873–880. MIT Press, Cambridge, MA, 2007.

O. L. Mangasarian. Generalized support vector machines. Technical report, University of Wisconsin, Computer Sciences Department, Madison, 1998.

B. D. McKay. nauty user's guide. Technical report, Dept. Computer Science, Austral. Nat. Univ., 1984.

A. Nobel and A. Dembo. A note on uniform laws of averages for dependent processes. *Statistics and Probability Letters*, 17:169–172, 1993.

N. Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23 (2):e177–e183, Jan 2007.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, Munich, 1997. Download: http://www.kernel-machines.org.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.

Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, A. Strehl, and S. V. N. Vishwanathan. Hash kernels. In Max Welling and David van Dyk, editors, *Proc. Intl. Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, 2009.

C. H. Teo and S. V. N. Vishwanathan. Fast and space efficient string kernels using suffix arrays. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 929–936, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-383-2. doi: http://doi.acm.org/10.1145/1143844.1143961.

H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma. Statistical evaluation of the predictive toxicology challenge 2000-2001. *Bioinformatics*, 19(10):1183–1193, July 2003.

K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18 (Suppl. 2):S268–S275, 2002.

S. V. N. Vishwanathan, Karsten Borgwardt, and Nicol N. Schraudolph. Fast computation of graph kernels. In B. Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*, Cambridge MA, 2007a. MIT Press.

S. V. N. Vishwanathan, A. J. Smola, and R. Vidal. Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. *International Journal of Computer Vision*, 73 (1):95–119, 2007b.

C. Watkins. Dynamic alignment kernels. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50, Cambridge, MA, 2000. MIT Press.

Journal of Machine Learning Research 10 (2009) 2639-2642 Submitted 5/09; Published 11/09

DL-Learner: Learning Concepts in Description Logics

Jens Lehmann LEHMANN@INFORMATIK.UNI-LEIPZIG.DE
Department of Computer Science
University of Leipzig
Johannisgasse 26, 04103 Leipzig, Germany

Editor: Soeren Sonnenburg

Abstract

In this paper, we introduce DL-Learner, a framework for learning in description logics and OWL. OWL is the official W3C standard ontology language for the Semantic Web. Concepts in this language can be learned for constructing and maintaining OWL ontologies or for solving problems similar to those in Inductive Logic Programming. DL-Learner includes several learning algorithms, support for different OWL formats, reasoner interfaces, and learning problems. It is a cross-platform framework implemented in Java. The framework allows easy programmatic access and provides a command line interface, a graphical interface as well as a WSDL-based web service.

Keywords: concept learning, description logics, OWL, classification, open-source

1. Introduction

The *Semantic Web* grows steadily[1] and contains knowledge from diverse areas such as science, music, literature, geography, social networks, as well as from upper and cross domain *ontologies*[2]. The underlying semantic technologies currently start to create substantial industrial impact in application scenarios on and off the web, including knowledge management, expert systems, web services, e-commerce, e-collaboration, etc. Since 2004, the *Web Ontology Language OWL*, which is based on *description logics* (Baader et al., 2007), has been the W3C-recommended standard for Semantic Web ontologies and is a key to the growth of the Semantic Web.

Within this field, there is a need for well-structured ontologies with large amounts of instance data, since engineering such ontologies constitutes a considerable investment of resources. Nowadays, knowledge bases often provide large amounts of instance data without sophisticated schemata. Methods for automated schema acquisition and maintenance are therefore sought (see, e.g., Buitelaar et al. 2007). In particular, concept learning methods have attracted much interest, see, for example, Esposito et al. (2004), Lehmann (2007), Lehmann and Hitzler (2008) and Lisi and Esposito (2008). DL-Learner provides an open source framework for such methods as we will briefly describe in the sequel. Several learning algorithms have been implemented within this framework. Outside of DL-Learner, there exist only non open source implementations of algorithms (YinYang, DL-FOIL) to the best of our knowledge.

1. To give a rough estimate, the semantic index Sindice (`http://sindice.com/`) lists more than 10 billion entities from more than 100 million web pages.
2. See, for example, `http://tomgruber.org/writing/ontology-definition-2007.htm` for a definition of ontology in computer science.

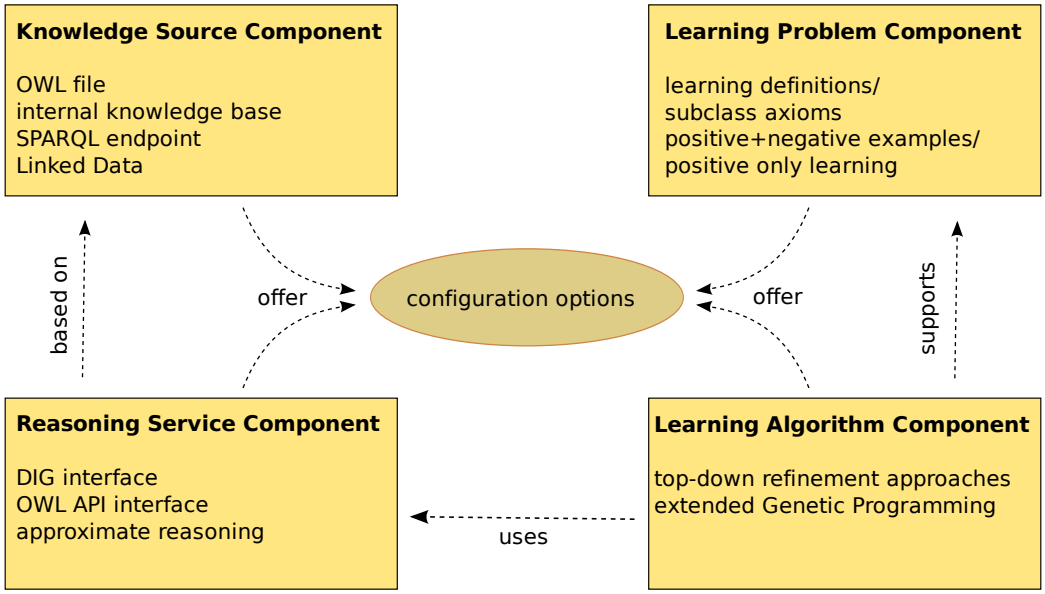


Figure 1: The architecture of DL-Learner is based on four component types each of which can have their own configuration options. A component manager can be used to create, combine, and configure components.

2. Framework

DL-Learner consists of core functionality, which provides Machine Learning algorithms for solving learning problems in OWL, support for different knowledge base formats, an OWL library, and reasoner interfaces. There are several interfaces for accessing this functionality, a couple of tools which use the DL-Learner algorithms, and a set of convenience scripts.

To be flexible and easily extensible, DL-Learner uses a component-based model. There are four types of components: knowledge source, reasoning service, learning problem, and learning algorithm. For each type, there are several implemented components and each component can have its own configuration options as illustrated in Figure 1. Configuration options can be used to change parameters/settings of a component.

Knowledge Sources integrate background knowledge. Almost all standard OWL formats are supported through the OWL API,[3] for example, RDF/XML, Manchester OWL Syntax, or Turtle. DL-Learner supports the inclusion of several knowledge sources, since knowledge can be widespread in the Semantic Web. In addition, DL-Learner facilitates the extraction of knowledge fragments from SPARQL[4] endpoints. This feature allows DL-Learner to scale up to very large knowledge bases containing millions of axioms (cf. Hellmann et al., 2009).

Reasoner Components provide connections to existing or own reasoners. Two components are the DIG 1.1[5] and OWL API reasoner interfaces, which allow to connect to all standard OWL reasoners via an HTTP and XML-based mechanism or a Java interface, respectively. Furthermore,

3. Information about the OWL API can be found at `http://owlapi.sourceforge.net`.
4. The W3C SPARQL recommendation is available at `http://www.w3.org/TR/rdf-sparql-query/`.
5. Information about DIG can be found at `http://dl.kr.org/dig/`.

DL-Learner offers its own approximate reasoner, which uses Pellet[6] for bootstrapping and loading the inferred model in memory. Afterwards, instance checks are performed very efficiently by using a local closed world assumption (see Badea and Nienhuys-Cheng 2000 on why this assumption is useful in description logics).

Learning Problems specify the problem type, which is to be solved by an algorithm. Currently, three problem components are implemented: 1.) learning from positive and negative examples 2.) positive-only learning and 3.) class axiom learning. The latter type is split into learning definitions and super class axioms. Amongst other methods, the components provide efficient coverage checks which can be used in the learning algorithms, for example, stochastic approaches for computing coverage up to a desired accuracy with respect to a 95% confidence interval are available.

Learning Algorithm components provide methods to solve one or more specified learning problem types. Apart from simple algorithms involving brute force or random guessing techniques, DL-Learner comprises a number of sophisticated algorithms based on genetic programming with a novel genetic operator (Lehmann, 2007), refinement operators for the description logic $\mathcal{ALC}$ (Lehmann and Hitzler, 2008), an extended operator supporting many features of OWL including datatype support, and an algorithm tailored for ontology engineering with a strong bias on short and readable concepts. Some of those algorithms have shown to be superior to other description logic learning systems and also superior to state-of-the-art ILP systems, for example, on the carcinogenesis problem.[7]

3. Implementation

The homepage of DL-Learner is `http://dl-learner.org` and contains up-to-date information about documentation and development of the software. A manual,[8] which complements the homepage and describes how to run DL-Learner, is included in its release. For developers, the Javadoc of DL-Learner is available online.[9]

The code base of DL-Learner consists of approximately 50,000 lines of code (excluding comments) with its core, that is, the component framework itself, accounting for roughly 1,500 lines. It is licensed under GPL 3. About 20 learning examples are included in the latest release (to be precise: 132 if smaller variations of existing problems/configurations are counted). 27 unit tests based on the JUnit framework are used to detect errors.

There are several interfaces available to access DL-Learner: To use components programmatically, the core package, in particular the component manager, can be of service. Similar methods are also available at the web service interface, which is based on WSDL. DL-Learner starts a web service included in Java 6, that is, no further tools are necessary. For end users, a command line interface is available. Settings are stored in *conf files*, which can then be executed in a similar fashion to other ILP tools. A prototypical graphical user interface is equally available, which can create, load, and save conf files. It provides widgets for modifying components and configuration options. An advantage of the component-based architecture is that all the interfaces mentioned need not to be changed, when new components are added or existing ones modified. This makes DL-Learner easily extensible. Another means to access DL-Learner, in particular for ontology engineering, is

6. The Pellet homepage is `http://clarkparsia.com/pellet/`.
7. `http://dl-learner.org/wiki/Carcinogenesis` presents benchmark results.
8. The DL-Learner manual is available at `http://dl-learner.org/files/dl-learner-manual.pdf`.
9. The DL-Learner Javadoc is available at `http://dl-learner.org/javadoc/`.

through plugins for the ontology editors OntoWiki[10] and Protégé.[11] The OntoWiki plugin is under construction, but can be used in its latest SVN version. The Protégé 4 plugin is included in the official Protégé plugin repository, that is, it is easy to install within Protégé.

Acknowledgments

I wish to acknowledge support by the OntoWiki EU FP7 project. Special thanks goes to Francesca Lisi for her comments, as well as the developers working on DL-Learner and tools based on it. Acknowledgements particularly to Sebastian Hellmann who worked on the SPARQL component, to Christoph Haase for his work on an EL refinement operator, to Christian Kötteritzsch for his work on the Protégé plugin, to Sebastian Bader who contributed a Prolog parser, and to those people using DL-Learner in their software.

References

F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2007. Cambridge University Press. ISBN 0-521-78176-0.

L. Badea and S.-H. Nienhuys-Cheng. A refinement operator for description logics. In *Proc. of the 10th Int. Conf. on Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 40–59. Springer, 2000.

P. Buitelaar, P. Cimiano, and B. Magnini, editors. *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence*. IOS, 2007.

F. Esposito, N. Fanizzi, L. Iannone, I. Palmisano, and G. Semeraro. Knowledge-intensive induction of terminologies from metadata. In *Proc. of 3rd Int. Semantic Web Conf.*, pages 441–455. Springer, 2004.

S. Hellmann, J. Lehmann, and S. Auer. Learning of OWL class descriptions on very large knowledge bases. *International Journal On Semantic Web and Information Systems, Special Issue on Scalability and Performance of Semantic Web Systems*, 5:25–48, 2009.

J. Lehmann. Hybrid learning of ontology classes. In *Proc. of 5th Int. Conf. on Machine Learning and Data Mining in Pattern Recognition*, volume 4571 of *Lecture Notes in Computer Science*, pages 883–898. Springer, 2007.

J. Lehmann and P. Hitzler. A refinement operator based learning algorithm for the ALC description logic. In *Proc. of 17th Int. Conf. on Inductive Logic Programming*, volume 4894 of *Lecture Notes in Computer Science*, pages 147–160. Springer, 2008. Awarded.

F.A. Lisi and F. Esposito. Foundations of onto-relational learning. In *Proc. of 18th Int. Conf. on Inductive Logic Programming*, volume 5194 of *Lecture Notes in Computer Science*, pages 158–175. Springer, 2008.

10. The OntoWiki homepage is `http://ontowiki.net`.
11. The Protégé homepage is `http://protege.stanford.edu`.

Journal of Machine Learning Research 10 (2009) 2643-2666 Submitted 12/08; Revised 9/09; Published 11/09

Bounded Kernel-Based Online Learning*

Francesco Orabona FORABONA@IDIAP.CH
Joseph Keshet† JKESHET@IDIAP.CH
Barbara Caputo BCAPUTO@IDIAP.CH
Idiap Research Institute
CH-1920 Martigny, Switzerland

Editor: Yoav Freund

Abstract

A common problem of kernel-based online algorithms, such as the kernel-based Perceptron algorithm, is the amount of memory required to store the online hypothesis, which may increase without bound as the algorithm progresses. Furthermore, the computational load of such algorithms grows linearly with the amount of memory used to store the hypothesis. To attack these problems, most previous work has focused on discarding some of the instances, in order to keep the memory bounded. In this paper we present a new algorithm, in which the instances are not discarded, but are instead projected onto the space spanned by the previous online hypothesis. We call this algorithm Projectron. While the memory size of the Projectron solution cannot be predicted before training, we prove that its solution is guaranteed to be bounded. We derive a relative mistake bound for the proposed algorithm, and deduce from it a slightly different algorithm which outperforms the Perceptron. We call this second algorithm Projectron++. We show that this algorithm can be extended to handle the multiclass and the structured output settings, resulting, as far as we know, in the first online bounded algorithm that can learn complex classification tasks. The method of bounding the hypothesis representation can be applied to any conservative online algorithm and to other online algorithms, as it is demonstrated for ALMA_2. Experimental results on various data sets show the empirical advantage of our technique compared to various bounded online algorithms, both in terms of memory and accuracy.

Keywords: online learning, kernel methods, support vector machines, bounded support set

1. Introduction

Kernel-based discriminative online algorithms have been shown to perform very well on binary and multiclass classification problems (see, for example, Freund and Schapire, 1999; Crammer and Singer, 2003; Kivinen et al., 2004; Crammer et al., 2006). Each of these algorithms works in rounds, where at each round a new instance is provided. On rounds where the online algorithm makes a prediction mistake or when the confidence in the prediction is not sufficient, the algorithm adds the instance to a set of stored instances, called the *support set*. The online classification function is defined as a weighted sum of kernel combination of the instances in the support set. It is clear that if the problem is not linearly separable or the target hypothesis is changing over time, the classification function will never stop being updated, and consequently, the support set will grow unboundedly.

*. A preliminary version of this paper appeared in the Proceedings of the 25th International Conference on Machine Learning under the title "The Projectron: a Bounded Kernel-Based Perceptron".

†. Current affiliation: Toyota Technological Institute at Chicago, 6045 S Kenwood Ave., Chicago, IL 60637.

This leads, eventually, to a memory explosion, which limits the applicability of these algorithms for those tasks, such as autonomous agents, for example, where data must be acquired continuously over time.

Several authors have tried to address this problem, mainly by bounding a priori the size of the support set with a fixed value, called a *budget*. The first algorithm to overcome the unlimited growth of the support set was proposed by Crammer et al. (2003), and refined by Weston et al. (2005). In these algorithms, once the size of the support set reaches the budget, an instance from the support set that meets some criterion is removed, and replaced by the new instance. The strategy is purely heuristic and no mistake bound is given. A similar strategy is also used in NORMA (Kivinen et al., 2004) and SILK (Cheng et al., 2007). The very first online algorithm to have a fixed memory budget and a relative mistake bound is the Forgetron algorithm (Dekel et al., 2007). A stochastic algorithm that on average achieves similar performance to Forgetron, and with a similar mistake bound was proposed by Cesa-Bianchi et al. (2006). Unlike all previous work, the analysis presented in the last paper is within a probabilistic context, and all the bounds derived there are in expectation. A different approach to address this problem for online Gaussian processes is proposed in Csató and Opper (2002), where, in common with our approach, the instances are not discarded, but rather projected onto the space spanned by the instances in the support set. However, in that paper no mistake bound is derived and there is no use of the hinge loss, which often produces sparser solutions. Recent work by Langford et al. (2008) proposed a parameter that trades accuracy for sparseness in the weights of online learning algorithms. Nevertheless, this approach cannot induce sparsity in online algorithms with kernels.

In this paper we take a different route. While previous work focused on discarding some of the instances in order to keep the support set bounded, in this work the instances are not discarded. Either they are projected onto the space spanned by the support set, or they are added to the support set. By using this method, we show that the support set and, hence, the online hypothesis, is guaranteed to be bounded, although we cannot predict its size before training. Instead of using a budget parameter, representing the maximum size of the support set, we introduce a parameter trading accuracy for sparseness, depending on the needs of the task at hand. The main advantage of this setup is that by using all training samples, we are able to provide an online hypothesis with high online accuracy. Empirically, as suggested by the experiments, the output hypotheses are represented with relatively small number of instances, and have high accuracy.

We start with the most simple and intuitive kernel-based algorithm, namely the kernel-based Perceptron. We modify the Perceptron algorithm so that the number of stored samples needed to represent the online hypothesis is always bounded. We call this new algorithm *Projectron*. The empirical performance of the Projectron algorithm is on a par with the original Perceptron algorithm. We present a relative mistake bound for the Projectron algorithm, and deduce from it a new online bounded algorithm which outperforms the Perceptron algorithm, but still retains all of its advantages. We call this second algorithm *Projectron++*. We then extend Projectron++ to the more general cases of multiclass and structured output. As far as we know, this is the first bounded multiclass and structured output online algorithm, with a relative mistake bound.[1] Our technique for bounding the size of the support set can be applied to any conservative kernel-based online algorithm and to other online algorithms, as we demonstrate for ALMA_2 (Gentile, 2001). Finally, we present some experiments with common data sets, which suggest that Projectron is comparable to

1. Note that converting the budget algorithms presented by other authors, such as the Forgetron, to the multiclass or the structured output setting is not trivial, since these algorithms are inherently binary in nature.

Perceptron in performance, but it uses a much smaller support set. Moreover, experiments with Projectron++ shows that it outperforms all other bounded algorithms, while using the smallest support set. We also present experiments on the task of phoneme classification, which is considered to be difficult and naturally with a relatively very high number of support vectors. When comparing the Projectron++ algorithm to the Passive-Aggressive multiclass algorithm (Crammer et al., 2006), it turns out that the cumulative online error and the test error, after online-to-batch conversion, of both algorithms are comparable, although Projectron++ uses a smaller support set.

In summary, the contributions of this paper are (1) a new algorithm, called Projectron, which is derived from the kernel-based Perceptron algorithm, which empirically performs equally well, but has a bounded support set; (2) a relative mistake bound for this algorithm; (3) another algorithm, called Projectron++, based on the notion of large margin, which outperforms the Perceptron algorithm and the proposed Projectron algorithm; (4) the multiclass and structured output Projectron++ online algorithm with a bounded support set; and (5) an extension of our technique to other online algorithms, exemplified in this paper for ALMA_2.

The rest of the paper is organized as follows: in Section 2 we state the problem definition and the kernel-based Perceptron algorithm. Section 3 introduces Projectron, along with its theoretical analysis. Next, in Section 4 we derive Projectron++. Section 5 presents the multiclass and structured learning variant of Projectron++. In Section 6 we apply our technique for another kernel-based online algorithm, ALMA_2. Section 7 describes experimental results of the algorithms presented, on different data sets. Section 8 concludes the paper with a short discussion.

2. Problem Setting and the Kernel-Based Perceptron Algorithm

The basis of our study is the well known *Perceptron* algorithm (Rosenblatt, 1958; Freund and Schapire, 1999). The Perceptron algorithm learns the mapping $f : \mathcal{X} \to \mathbb{R}$ based on a set of examples $\mathcal{T} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots\}$, where $\mathbf{x}_t \in \mathcal{X}$ is called an *instance* and $y_t \in \{-1, +1\}$ is called a *label*. We denote the prediction of the Perceptron algorithm as $\text{sign}(f(\mathbf{x}))$ and we interpret $|f(\mathbf{x})|$ as the confidence in the prediction. We call the output f of the Perceptron algorithm a *hypothesis*, and we denote the set of all attainable hypotheses by $\mathcal{H}$. In this paper we assume that $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) with a positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ implementing the inner product $\langle \cdot, \cdot \rangle$. The inner product is defined so that it satisfies the reproducing property, $\langle k(\mathbf{x}, \cdot), f(\cdot) \rangle = f(\mathbf{x})$.

The Perceptron algorithm is an online algorithm, where learning takes place in rounds. At each round a new hypothesis function is estimated, based on the previous one. We denote the hypothesis estimated after the t-th round by f_t. The algorithm starts with the zero hypothesis, $f_0 = \mathbf{0}$. At each round t, an instance $\mathbf{x}_t \in \mathcal{X}$ is presented to the algorithm, which predicts a label $\hat{y}_t \in \{-1, +1\}$ using the current function, $\hat{y}_t = \text{sign}(f_t(\mathbf{x}_t))$. Then, the correct label y_t is revealed. If the prediction $\hat{y}_t$ differs from the correct label y_t, the hypothesis is updated as $f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$, otherwise the hypothesis is left intact, $f_t = f_{t-1}$. The hypothesis f_t can be written as a kernel expansion according to the representer theorem (Schölkopf et al., 2001),

$$f_t(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{S}_t} \alpha_i k(\mathbf{x}_i, \mathbf{x}), \tag{1}$$

where $\alpha_i = y_i$ and $\mathcal{S}_t$ is defined to be the set of instances for which an update of the hypothesis occurred, that is, $\mathcal{S}_t = \{\mathbf{x}_i, 0 \le i \le t \,|\, \hat{y}_i \ne y_i\}$. The set $\mathcal{S}_t$ is called the *support set*. The Perceptron algorithm is summarized in Figure 1.

Initialize: $\mathcal{S}_0 = \emptyset$, $f_0 = \mathbf{0}$
For $t = 1, 2, \ldots$
  Receive new instance $\mathbf{x}_t$
  Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$
  Receive label y_t
  If $y_t \neq \hat{y}_t$
    $f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$ *(update the hypothesis)*
    $\mathcal{S}_t = \mathcal{S}_{t-1} \cup \mathbf{x}_t$ *(add instance $\mathbf{x}_t$ to the support set)*
  Else
    $f_t = f_{t-1}$
    $\mathcal{S}_t = \mathcal{S}_{t-1}$

Figure 1: The kernel-based Perceptron Algorithm.

Although the Perceptron algorithm is very simple, it produces an online hypothesis with good performance. Our goal is to derive and analyze a new algorithm, which outputs a hypothesis that attains almost the same performance as the Perceptron hypothesis, but can be represented using many fewer instances, that is, an online hypothesis that is "close" to the Perceptron hypothesis but represented by a smaller support set. Recall that the hypothesis f_t is represented as a weighted sum over all the instances in the support set. The size of this representation is the cardinality of the support set, $|\mathcal{S}_t|$.

3. The Projectron Algorithm

This section starts by deriving the Projectron algorithm, motivated by an example of a finite dimensional kernel space. It continues with a description of how to calculate the projected hypothesis and describes some other computational aspects of the algorithm. The section concludes with a theoretical analysis of the algorithm.

3.1 Definition and Derivation

Let us first consider a finite dimensional RKHS $\mathcal{H}$ induced by a kernel such as the polynomial kernel. Since $\mathcal{H}$ is finite dimensional, there are a finite number of linearly independent hypotheses in this space. Hence, any hypothesis in this space can be expressed using a finite number of examples. We can modify the Perceptron algorithm to use only one set of independent instances as follows. On each round the algorithm receives an instance and predicts its label. On a prediction mistake, we check if the instance $\mathbf{x}_t$ can be spanned by the support set, namely, for scalars $d_i \in \mathbb{R}, 1 \leq i \leq |\mathcal{S}_{t-1}|$, not all zeros, such that

$$k(\mathbf{x}_t, \cdot) = \sum_{\mathbf{x}_i \in \mathcal{S}_{t-1}} d_i k(\mathbf{x}_i, \cdot) \ .$$

If we can find such scalars, the instance is not added to the support set, but instead, the coefficients $\{\alpha_i\}$ in the expansion Equation (1) are changed to reflect the addition of this instance to the support

set. Namely, for every i

$$\alpha_i = y_i + y_t d_i \ .$$

On the other hand, if the instance and the support set are linearly independent, the instance is added to the set with $\alpha_t = y_t$ as before. This technique reduces the size of the support set without changing the hypothesis. A similar approach was used by Downs et al. (2001) to simplify SVM solutions.

Let us consider now the more elaborate case of an infinite dimensional RKHS $\mathcal{H}$ induced by a kernel such as the Gaussian kernel. In this case, it is not possible to find a finite number of linearly independent vectors which span the whole space, and hence there is no guarantee that the hypothesis can be expressed by a finite number of instances. However, we can approximate the concept of linear independence with a finite number of vectors (Csató and Opper, 2002; Engel et al., 2004; Orabona et al., 2007).

In particular, let us assume that at round t of the algorithm there is a prediction mistake, and that the mistaken instance $\mathbf{x}_t$ should be added to the support set, $\mathcal{S}_{t-1}$. Let $\mathcal{H}_{t-1}$ be an RKHS which is the span of the kernel images of the instances in the set $\mathcal{S}_{t-1}$. Formally,

$$\mathcal{H}_{t-1} = \text{span}\left(\{k(\mathbf{x},\cdot)|\mathbf{x} \in \mathcal{S}_{t-1}\}\right) \ . \tag{2}$$

Before adding the instance to the support set, we construct two hypotheses: a *temporary hypothesis*, f'_t, using the function $k(\mathbf{x}_t,\cdot)$, that is, $f'_t = f_{t-1} + y_t k(\mathbf{x}_t,\cdot)$, and a *projected hypothesis*, f''_t, that is the projection of f'_t onto the space $\mathcal{H}_{t-1}$. That is, the projected hypothesis is that hypothesis from the space $\mathcal{H}_{t-1}$ which is the *closest* to the temporary hypothesis. In a later section we will describe an efficient way to calculate the projected hypothesis. Denote by δ_t the distance between the hypotheses, $\delta_t = f''_t - f'_t$. If the norm of distance $\|\delta_t\|$ is below some threshold η, we use the projected hypothesis as our next hypothesis, that is, $f_t = f''_t$, otherwise we use the temporary hypothesis as our next hypothesis, that is, $f_t = f'_t$. As we show in the following theorem, this strategy assures that the maximum size of the support set is always finite, regardless of the dimension of the RKHS $\mathcal{H}$. Guided by these considerations we can design a new Perceptron-like algorithm that projects the solution onto the space spanned by the previous support vectors whenever possible. We call this algorithm *Projectron*. The algorithm is given in Figure 2.

The parameter η plays an important role in our algorithm. If η is equal to zero, we obtain exactly the same solution as the Perceptron algorithm. In this case, however, the Projectron solution can still be sparser when some of the instances are linearly dependent or when the kernel induces a finite dimensional RKHS $\mathcal{H}$. If η is greater than zero we trade precision for sparseness. Moreover, as shown in the next section, this implies a bounded algorithmic complexity, namely, the memory and time requirements for each step are bounded. We analyze the effect of η on the classification accuracy in Subsection 3.3.

3.2 Practical Considerations

We now consider the problem of deriving the projected hypothesis f''_t in a Hilbert space $\mathcal{H}$, induced by a kernel function $k(\cdot,\cdot)$. Recall that f'_t is defined as $f'_t = f_t + y_t k(\mathbf{x}_t,\cdot)$. Denote by $P_{t-1} f'_t$ the projection of $f'_t \in \mathcal{H}$ onto the subspace $\mathcal{H}_{t-1} \subseteq \mathcal{H}$. The projected hypothesis f''_t is defined as $f''_t = P_{t-1} f'_t$. Schematically, this is depicted in Figure 3.

Expanding f'_t we have

$$f''_t = P_{t-1} f'_t = P_{t-1}\left(f_{t-1} + y_t k(\mathbf{x}_t,\cdot)\right) \ .$$

Initialize: $\mathcal{S}_0 = \emptyset$, $f_0 = \mathbf{0}$
For $t = 1, 2, \ldots$
  Receive new instance $\mathbf{x}_t$
  Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$
  Receive label y_t
  If $y_t \neq \hat{y}_t$
    Set $f'_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$ (*temporary hypothesis*)
    Set $f''_t = f'_t$ projected onto the space $\mathcal{H}_{t-1}$ (*projected hypothesis*)
    Set $\delta_t = f''_t - f'_t$
    If $\|\delta_t\| \leq \eta$
      $f_t = f''_t$
      $\mathcal{S}_t = \mathcal{S}_{t-1}$
    Else
      $f_t = f'_t$
      $\mathcal{S}_t = \mathcal{S}_{t-1} \cup \mathbf{x}_t$ (*add* $\mathbf{x}_t$ *to the support set)*
  Else
    $f_t = f_{t-1}$
    $\mathcal{S}_t = \mathcal{S}_{t-1}$

Figure 2: The Projectron Algorithm.

The projection is a linear operator, hence

$$f''_t = f_{t-1} + y_t P_{t-1} k(\mathbf{x}_t, \cdot) \; . \tag{3}$$

Recall that $\delta_t = f''_t - f'_t$. By substituting f''_t from Equation (3) and f'_t we have

$$\delta_t = f''_t - f'_t = y_t P_{t-1} k(\mathbf{x}_t, \cdot) - y_t k(\mathbf{x}_t, \cdot) \; . \tag{4}$$

The projection of $f'_t \in \mathcal{H}$ onto a subspace $\mathcal{H}_{t-1} \subset \mathcal{H}$ is defined as the hypothesis in $\mathcal{H}_{t-1}$ closest to f'_t. Hence, let $\sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \cdot)$ be an hypothesis in $\mathcal{H}_{t-1}$, where $\mathbf{d} = (d_1, \ldots, d_{|\mathcal{S}_{t-1}|})$ is a set of coefficients, with $d_i \in \mathbb{R}$. The closest hypothesis is the one for which it holds that

$$\|\delta_t\|^2 = \min_{\mathbf{d}} \left\| \sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \cdot) - k(\mathbf{x}_t, \cdot) \right\|^2 \; . \tag{5}$$

Expanding Equation (5) we get

$$\|\delta_t\|^2 = \min_{\mathbf{d}} \left(\sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{S}_{t-1}} d_j d_i k(\mathbf{x}_j, \mathbf{x}_i) - 2 \sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \mathbf{x}_t) + k(\mathbf{x}_t, \mathbf{x}_t) \right) \; .$$

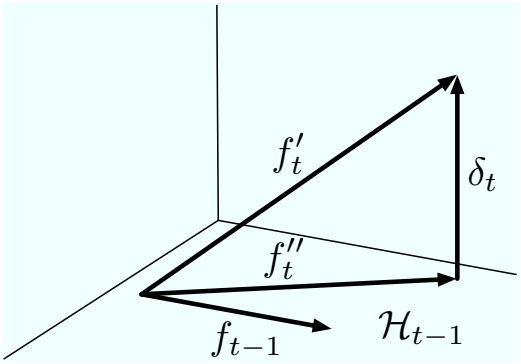


Figure 3: Geometrical interpretation of the projection of the hypothesis f''_t onto the subspace $\mathcal{H}_{t-1}$.

Let us define $\mathbf{K}_{t-1} \in \mathbb{R}^{t-1 \times t-1}$ to be the matrix generated by the instances in the support set $\mathcal{S}_{t-1}$, that is, $\{\mathbf{K}_{t-1}\}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ for every $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{S}_{t-1}$. Let us also define $\mathbf{k}_t \in \mathbb{R}^{t-1}$ to be the vector whose i-th element is $k_{t_i} = k(\mathbf{x}_i, \mathbf{x}_t)$. We have

$$\|\delta_t\|^2 = \min_{\mathbf{d}} \left(\mathbf{d}^T \mathbf{K}_{t-1} \mathbf{d} - 2\mathbf{d}^T \mathbf{k}_t + k(\mathbf{x}_t, \mathbf{x}_t) \right) . \tag{6}$$

Solving Equation (6), that is, applying the extremum conditions with respect to $\mathbf{d}$, we obtain

$$\mathbf{d}^\star = \mathbf{K}_{t-1}^{-1} \mathbf{k}_t \tag{7}$$

and, by substituting Equation (7) into Equation (6),

$$\|\delta_t\|^2 = k(\mathbf{x}_t, \mathbf{x}_t) - \mathbf{k}_t^T \mathbf{d}^\star . \tag{8}$$

Furthermore, by substituting Equation (7) back into Equation (3) we get

$$f''_t = f_{t-1} + y_t \sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} \mathbf{d}^\star_j k(\mathbf{x}_j, \cdot) . \tag{9}$$

We have shown how to calculate both the distance δ_t and the projected hypothesis f''_t. In summary, one needs to compute $\mathbf{d}^\star$ according to Equation (7), and plug the result either into Equation (8) to obtain δ_t, or into Equation (9) to obtain the projected hypothesis.

In order to make the computation more tractable, we need an efficient method to calculate the matrix inversion $\mathbf{K}_t^{-1}$ iteratively. The first method, used by Cauwenberghs and Poggio (2000) for incremental training of SVMs, directly updates the inverse matrix. An efficient way to do this, exploiting the incremental nature of the approach, is to recursively update the inverse matrix. Using the matrix inversion lemma it is possible to show (see, e.g., Csató and Opper, 2002) that after the addition of a new sample, $\mathbf{K}_t^{-1}$ becomes

$$\mathbf{K}_t^{-1} = \begin{bmatrix} & & & 0 \\ & \mathbf{K}_{t-1}^{-1} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\|\delta_t\|^2} \begin{bmatrix} \mathbf{d}^\star \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{d}^{\star T} & -1 \end{bmatrix}$$

Input: new instance $\mathbf{x}_t$, $\mathbf{K}_{t-1}^{-1}$, and the support set $\mathcal{S}_{t-1}$

- Set $\mathbf{k}_t = \big(k(\mathbf{x}_1,\mathbf{x}_t), k(\mathbf{x}_2,\mathbf{x}_t), \ldots, k(\mathbf{x}_{|\mathcal{S}_{t-1}|},\mathbf{x}_t)\big)$
- Solve $\mathbf{d}^\star = \mathbf{K}_{t-1}^{-1}\mathbf{k}_t$
- Set $\|\delta_t\|^2 = k(\mathbf{x}_t,\mathbf{x}_t) - \mathbf{k}_t^T\mathbf{d}^\star$
- The projected hypothesis is $f_t'' = f_{t-1} + y_t \sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} \mathbf{d}_j^\star k(\mathbf{x}_j,\cdot)$
- Kernel inverse matrix for the next round

$$\mathbf{K}_t^{-1} = \begin{bmatrix} & & & 0 \\ & \mathbf{K}_{t-1}^{-1} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\|\delta_t\|^2} \begin{bmatrix} \mathbf{d}^\star \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{d}^{\star T} & -1 \end{bmatrix}$$

Output: the projected hypothesis f_t'', the measure δ_t and the kernel inverse matrix $\mathbf{K}_t^{-1}$.

Figure 4: Calculation of the projected hypothesis f_t''.

where $\mathbf{d}^\star$ and $\|\delta_t\|^2$ are already evaluated during the previous steps of the algorithm, as given by Equation (7) and Equation (8). Thanks to this incremental evaluation, the time complexity of the linear independence check is $O(|\mathcal{S}_{t-1}|^2)$, as one can easily see from Equation (7). Note that the matrix $\mathbf{K}_{t-1}$ can be safely inverted since, by incremental construction, it is always full-rank.

An alternative way to derive the inverse matrix is to use the Cholesky decomposition of $\mathbf{K}_{t-1}$ and to update it recursively. This is known to be numerically more stable than directly updating the inverse. In our experiments, however, we found out that the method presented here is as stable as the Cholesky decomposition.

Overall, the time complexity of the algorithm is $O(|\mathcal{S}_t|^2)$, as described above, and the space complexity is $O(|\mathcal{S}_t|^2)$, due to the storage of the matrix $\mathbf{K}_t^{-1}$, similar to the second-order Perceptron algorithm (Cesa-Bianchi et al., 2005). A summary of the derivation of f_t'', the projection of f_t' onto the space spanned by $\mathcal{S}_{t-1}$, is described in Figure 4.

3.3 Analysis

We now analyze the theoretical aspects of the proposed algorithm. First, we present a theorem which states that the size of the support set of the Projectron algorithm is bounded.

Theorem 1 *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a continuous Mercer kernel, with $\mathcal{X}$ a compact subset of a Banach space. Then, for any training sequence $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}, i = 1, 2, \cdots$ and for any $\eta > 0$, the size of the support set of the Projectron algorithm is finite.*

Proof The proof of this theorem follows the same lines as the proof of Theorem 3.1 in Engel et al. (2004). From the Mercer theorem it follows that there exists a mapping $\phi : \mathcal{X} \to \mathcal{H}'$, where $\mathcal{H}'$ is

an Hilbert space, $k(\mathbf{x},\mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ and ϕ is continuous. Given that ϕ is continuous and that $\mathcal{X}$ is compact, we obtain that $\phi(\mathcal{X})$ is compact. From the definition of δ_t in Equation (5) we get that every time a new basis vector is added we have

$$\begin{aligned}\eta^2 \leq \|\delta_t\|^2 &= \min_{\mathbf{d}} \left\| \sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \cdot) - k(\mathbf{x}_t, \cdot) \right\|^2 \leq \min_{d_j} \left\| d_j k(\mathbf{x}_j, \cdot) - k(\mathbf{x}_t, \cdot) \right\|^2 \\ &= \min_{d_j} \left\| d_j \phi(\mathbf{x}_j) - \phi(\mathbf{x}_t) \right\|^2 \leq \left\| \phi(\mathbf{x}_j) - \phi(\mathbf{x}_t) \right\|^2\end{aligned}$$

for any $1 \leq j \leq |\mathcal{S}_{t-1}|$. Hence from the definition of packing numbers (Cucker and Zhou, 2007, Definition 5.17), we get that the maximum size of the support set in the Projectron algorithm is bounded by the packing number at scale η of $\phi(\mathcal{X})$. This number, in turn, is bounded by the covering number at scale $\eta/2$, and it is finite because the set is compact (Cucker and Zhou, 2007, Proposition 5.18). ■

Note that this theorem guarantees that the size of the support set is bounded, however it does not state that the size of the support set is fixed or that it can be estimated before training.

The next theorem provides a mistake bound. The main idea is to bound the maximum number of mistakes of the algorithm, relative to any hypothesis $g \in \mathcal{H}$, even chosen in hindsight. First, we define the loss with a margin $\gamma \in \mathbb{R}$ of the hypothesis g on the example $(\mathbf{x}_t, y_t)$ as

$$\ell_\gamma(g(\mathbf{x}_t), y_t) = \max\{0, \gamma - y_t g(\mathbf{x}_t)\}, \tag{10}$$

and we define the cumulative loss, D_γ, of g on the first T examples as

$$D_\gamma = \sum_{t=1}^{T} \ell_\gamma(g(\mathbf{x}_t), y_t) \ .$$

Before stating the bound, we present a lemma that will be used in the rest of our proofs. We will use its first statement to bound the scalar product between a projected sample and the competitor, and its second statement to derive the scalar product between the current hypothesis and the projected sample.

Lemma 2 *Let $(\hat{\mathbf{x}}, \hat{y})$ be an example, with $\hat{\mathbf{x}} \in \mathcal{X}$ and $\hat{y} \in \{+1, -1\}$. If we denote by $f(\cdot)$ an hypothesis in $\mathcal{H}$, and denote by $q(\cdot)$ any function in $\mathcal{H}$, then the following holds*

$$\hat{y}\langle f, q \rangle \geq \gamma - \ell_\gamma(f(\hat{\mathbf{x}}), \hat{y}) - \|f\| \cdot \|q - k(\hat{\mathbf{x}}, \cdot)\| \ .$$

Moreover, if $f(\cdot)$ can be written as $\sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \cdot)$ with $\alpha_i \in \mathbb{R}$ and $\mathbf{x}_i \in \mathcal{X}, i = 1, \cdots, m$, and $q(\cdot)$ is the projection of $k(\hat{\mathbf{x}}, \cdot)$ onto the space spanned by $k(\mathbf{x}_i, \cdot), i = 1, \cdots, m$, then

$$\hat{y}\langle f, q \rangle = \hat{y} f(\hat{\mathbf{x}}) \ .$$

Proof The first inequality comes from an application of the Cauchy-Schwarz inequality and the definition of the hinge loss in Equation (10). The second equality follows from the fact that $\langle f, q - k(\hat{\mathbf{x}}, \cdot) \rangle = 0$, because $f(\cdot)$ is orthogonal to the difference between $k(\hat{\mathbf{x}}, \cdot)$ and its projection onto the space in which $f(\cdot)$ lives. ■

With these definitions at hand, we can state the following bound for Projectron.

Theorem 3 *Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be a sequence of instance-label pairs where $\mathbf{x}_t \in \mathcal{X}$, $y_t \in \{-1, +1\}$, and $k(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Assume that the Projectron algorithm is run with $\eta \geq 0$. Then the number of prediction mistakes it makes on the sequence is bounded by*

$$\frac{\|g\|^2}{(1-\eta\|g\|)^2} + \frac{D_1}{1-\eta\|g\|} + \frac{\|g\|}{1-\eta\|g\|}\sqrt{\frac{D_1}{1-\eta\|g\|}}$$

where g is an arbitrary function in $\mathcal{H}$, such that $\|g\| < \frac{1}{\eta}$.

Proof Define the relative progress in each round as $\Delta_t = \|f_{t-1} - \lambda g\|^2 - \|f_t - \lambda g\|^2$, where λ is a positive scalar to be optimized. We bound the progress from above and below, as in Gentile (2003). On rounds where there is no mistake, Δ_t equals 0. On rounds where there is a mistake there are two possible updates: either $f_t = f_{t-1} + y_t P_{t-1} k(\mathbf{x}_t, \cdot)$ or $f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$. In the following we start bounding the progress from below, when the update is of the former type. In particular we set $q(\cdot) = P_{t-1} k(\mathbf{x}_t, \cdot)$ in Lemma 2 and use $\delta_t = y_t P_{t-1} k(\mathbf{x}_t, \cdot) - y_t k(\mathbf{x}_t, \cdot)$ from Equation (4). Let τ_t be an indicator function for a mistake on the t-th round, that is, τ_t is 1 if there is a mistake on round t and 0 otherwise. We have

$$\begin{aligned}\Delta_t &= \|f_{t-1} - \lambda g\|^2 - \|f_t - \lambda g\|^2 = 2\tau_t y_t \langle \lambda g - f_{t-1}, P_{t-1} k(\mathbf{x}_t, \cdot)\rangle - \tau_t^2 \|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2 \\ &\geq \tau_t \Big(2\lambda - 2\lambda \ell_1(g(\mathbf{x}_t), y_t) - \tau_t \|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2 - 2\lambda \|g\| \cdot \|\delta_t\| - 2 y_t f_{t-1}(\mathbf{x}_t)\Big) \,. \end{aligned} \tag{11}$$

Moreover, on every projection update $\|\delta_t\| \leq \eta$, and $\|P_{t-1} k(\mathbf{x}_t, \cdot)\| \leq 1$ by the theorem's assumption, so we have

$$\Delta_t \geq \tau_t \Big(2\lambda - 2\lambda \ell_1(g(\mathbf{x}_t), y_t) - \tau_t - 2\eta\lambda \|g\| - 2 y_t f_{t-1}(\mathbf{x}_t)\Big) \,.$$

We can further bound Δ_t by noting that on every prediction mistake $y_t f_{t-1}(\mathbf{x}_t) \leq 0$. Overall we have

$$\|f_{t-1} - \lambda g\|^2 - \|f_t - \lambda g\|^2 \geq \tau_t \Big(2\lambda - 2\lambda \ell_1(g(\mathbf{x}_t), y_t) - \tau_t - 2\eta\lambda \|g\|\Big) \,. \tag{12}$$

When there is an update without projection, similar reasoning yields that

$$\|f_{t-1} - \lambda g\|^2 - \|f_t - \lambda g\|^2 \geq \tau_t \Big(2\lambda - 2\lambda \ell_1(g(\mathbf{x}_t), y_t) - \tau_t\Big),$$

hence the bound in Equation (12) holds in both cases.

We sum over t on both sides, remembering that τ_t can be upper bounded by 1. The left hand side of the equation is a telescoping sum, hence it collapses to $\|f_0 - \lambda g\|^2 - \|f_T - \lambda g\|^2$, which can be upper bounded by $\lambda^2 \|g\|^2$, using the fact that $f_0 = \mathbf{0}$ and that $\|f_T - \lambda g\|^2$ is non-negative. Finally, we have

$$\lambda^2 \|g\|^2 + 2\lambda D_1 \geq M\left(2\lambda - 2\eta\lambda \|g\| - 1\right), \tag{13}$$

where M is the number of mistakes. The last equation implies a bound on M for any choice of $\lambda > 0$, hence we can take the minimum of these bounds. From now on we can suppose that $M > \frac{D_1}{1-\eta\|g\|}$. In fact, if $M \leq \frac{D_1}{1-\eta\|g\|}$ then the theorem trivially holds. The minimum of Equation (13) as a function of λ occurs at

$$\lambda^* = \frac{M(1-\eta\|g\|) - D_1}{\|g\|^2} \,.$$

By our hypothesis that $M > \frac{D_1}{1-\eta\|g\|}$ we have that λ^* is positive. Substituting λ^* into Equation (13) we obtain

$$\frac{(D_1 - M(1-\eta\|g\|))^2}{\|g\|^2} - M \leq 0 \ .$$

Solving for M and overapproximating concludes the proof. ■

This theorem suggests that the performance of the Projectron algorithm is slightly worse than that of the Perceptron algorithm. Specifically, if we set $\eta = 0$, we recover the best known bound for the Perceptron algorithm (see for example Gentile, 2003). Hence the degradation in the performance of Projectron compared to Perceptron is related to $\frac{1}{1-\eta\|g\|}$. Empirically, the Projectron algorithm and the Perceptron algorithm perform similarly, for a wide range of settings of η.

4. The Projectron++ Algorithm

The proof of Theorem 3 suggests how to improve the Projectron algorithm to improve upon the performance of the Perceptron algorithm, while maintaining a bounded support set. We can change the Projectron algorithm so that an update takes place not only if there is a prediction mistake, but also when the confidence of the prediction is low. We refer to this latter case as a *margin error*, that is, $0 < y_t f_{t-1}(\mathbf{x}_t) < 1$. This strategy is known to improve the classification rate but also increases the size of the support set (Crammer et al., 2006). A possible solution to this obstacle is not to update on every round in which a margin error occurs, but only when there is a margin error and the new instance *can be projected* onto the support set. Hence, the update on round in which there is a margin error would in general be of the form

$$f_t = f_{t-1} + y_t \tau_t P_{t-1} k(\mathbf{x}_t, \cdot) \ ,$$

with $0 < \tau_t \leq 1$. The last constraint comes from the proof of Theorem 3, where we upper bound τ_t by 1. Note that setting τ_t to 0 is equivalent to leaving the hypothesis unchanged.

In particular, disregarding the loss term in Equation (11), the progress Δ_t can be made positive with an appropriate choice of τ_t. Whenever this progress is non-negative the worst-case number of mistakes decreases, hopefully along with the classification error rate of the algorithm. With this modification we expect better performance, that is, fewer mistakes, but without any increase of the support set size. We can even expect solutions with a smaller support set, since new instances can be added to the support set only if misclassified, hence having fewer mistakes should result in a smaller support set. We name this algorithm *Projectron++*. The following theorem states a mistake bound for Projectron++, and guides us in how to choose τ_t.

Theorem 4 *Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be a sequence of instance-label pairs where $\mathbf{x}_t \in \mathcal{X}$, $y_t \in \{-1, +1\}$, and $k(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Assume that Projectron++ is run with $\eta > 0$. Then the number of prediction mistakes it makes on the sequence is bounded by*

$$\frac{\|g\|^2}{(1-\eta\|g\|)^2} + \frac{D_1}{1-\eta\|g\|} + \frac{\|g\|}{1-\eta\|g\|}\sqrt{max\left(0, \frac{D_1}{1-\eta\|g\|} - B\right)}$$

where g is an arbitrary function in $\mathcal{H}$, such that $\|g\| < \frac{1}{\eta}$,

$$0 < \tau_t < \min \left\{ 2\frac{\ell_1(f_{t-1}(\mathbf{x}_t), y_t) - \frac{\|\delta_t\|}{\eta}}{\|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2}, 1 \right\}$$

and

$$B = \sum_{\{t:0<y_t f_{t-1}(\mathbf{x}_t)<1\}} \tau_t \left(2\ell_1(f_{t-1}(\mathbf{x}_t), y_t) - \tau_t \|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2 - 2\frac{\|\delta_t\|}{\eta} \right) > 0 \, .$$

Proof The proof is similar to the proof of Theorem 3, where the difference is that during rounds in which there is a margin error we update the solution whenever it is possible to project ensuring an improvement of the mistake bound. Assume that $\lambda \geq 1$. On rounds when a margin error occurs, as in Equation (11), we can write

$$\begin{aligned}
\Delta_t + 2\tau_t\lambda\ell_1(g(\mathbf{x}_t), y_t) \geq & \quad \tau_t \left(2\lambda - \tau_t \|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2 - 2\lambda\|\delta_t\| \cdot \|g\| - 2y_t f_{t-1}(\mathbf{x}_t) \right) \\
> & \quad \tau_t \left(2\left(1 - \frac{\|\delta_t\|}{\eta}\right) - \tau_t \|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2 - 2y_t f_{t-1}(\mathbf{x}_t) \right) \\
= & \quad \tau_t \left(2\ell_1(f_{t-1}(\mathbf{x}_t), y_t) - \tau_t \|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2 - 2\frac{\|\delta_t\|}{\eta} \right), \qquad (14)
\end{aligned}$$

where we used the bounds on $\|g\|$ and λ. Let β_t be the right hand-side of Equation (14). A sufficient condition to have β_t positive is

$$\tau_t < 2\frac{\ell_1(f_{t-1}(\mathbf{x}_t), y_t) - \frac{\|\delta_t\|}{\eta}}{\|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2} \, .$$

Constraining τ_t to be less than or equal to 1 yields the update rule in the theorem.

Let $B = \sum_{\{t:0<y_t f_{t-1}(\mathbf{x}_t)<1\}} \beta_t$. Similarly to the proof of Theorem 3, we have

$$\lambda^2\|g\|^2 + 2\lambda D_1 \geq M(2\lambda - 2\eta\lambda\|g\| - 1) + B \, . \qquad (15)$$

Again, the optimal value of λ is

$$\lambda^* = \frac{M(1 - \eta\|g\|) - D_1}{\|g\|^2} \, .$$

We can assume that $M(1-\eta\|g\|) - D_1 \geq \|g\|^2$. In fact, if $M < \frac{\|g\|^2 + D_1}{1-\eta\|g\|}$, then the theorem trivially holds. With this assumption, λ^* is positive and greater than or equal to 1, satisfying our initial constraint on λ. Substituting this optimal value of λ into Equation (15), we have

$$\frac{(D_1 - M(1-\eta\|g\|))^2}{\|g\|^2} - M + B \leq 0 \, .$$

Solving for M concludes the proof. ■

The proof technique presented here is very general, in particular it can be applied to the Passive-Aggressive algorithm PA-I (Crammer et al., 2006). In fact, removing the projection step and updating on rounds in which there is a margin error, with $f_t = f_{t-1} + y_t \tau_t k(\mathbf{x}_t, \cdot)$, we end up with the condition $0 < \tau_t < \min\left\{2\frac{\ell_1(f_{t-1}(\mathbf{x}_t), y_t)}{\|k(\mathbf{x}_t,\cdot)\|^2}, 1\right\}$. This rule generalizes the PA-I bound whenever $R = 1$ and $C = 1$, however the obtained bound substantially improves upon the original bound in Crammer et al. (2006).

The theorem gives us some freedom for the choice of τ_t. Experimentally we have observed that we obtain the best performance if the update is done with the following rule

$$\tau_t = \min\left\{\frac{\ell_1(f_{t-1}(\mathbf{x}_t), y_t)}{\|P_{t-1}k(\mathbf{x}_t,\cdot)\|^2}, 2\frac{\ell_1(f_{t-1}(\mathbf{x}_t), y_t) - \frac{\|\delta_t\|}{\eta}}{\|P_{t-1}k(\mathbf{x}_t,\cdot)\|^2}, 1\right\}.$$

The added term in the minimum comes from ignoring the term $-2\frac{\|\delta_t\|}{\eta}$ and in finding the maximum of the quadratic equation. Notice that the term $\|P_{t-1}k(\mathbf{x}_t,\cdot)\|^2$ in the last equation can be practically computed as $\mathbf{k}_t^T \mathbf{d}^\star$, as can be derived using the same techniques presented in Subsection 3.2.

We note in passing that the condition on whether $\mathbf{x}_t$ can be projected onto $\mathcal{H}_{t-1}$ on margin error may stated as $\ell_1(f_{t-1}(\mathbf{x}_t), y_t) \geq \frac{\|\delta_t\|}{\eta}$. This means that if the loss is relatively large, the progress is also large and the algorithm can afford "wasting" a bit of it for the sake of projecting.

The algorithm is summarized in Figure 2. The performance of the Projectron++ algorithm, the Projectron algorithm and several other bounded online algorithms are compared and reported in Section 7.

5. Extension to Multiclass and Structured Output

In this section we extend Projectron++ to the multiclass and the structured output settings (note that Projectron can be generalized in a similar way). We start by presenting the more complex decision problem, namely the structured output, and then we derive the multiclass decision problem as a special case.

In structured output decision problems the set of possible labels has a unique and defined structure, such as a tree, a graph or a sequence (Collins, 2000; Taskar et al., 2003; Tsochantaridis et al., 2004). Denote the set of all labels as $\mathcal{Y} = \{1, \ldots, k\}$. Each instance is associated with a label from $\mathcal{Y}$. Generally, in structured output problems there may be dependencies between the instance and the label, as well as between labels. Hence, to capture these dependencies, the input and the output pairs are represented in a common feature representation. The learning task is therefore defined as finding a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ such that

$$y_t = \arg\max_{y \in Y} f(\mathbf{x}_t, y)\ . \tag{16}$$

Let us generalize the definition of the RKHS $\mathcal{H}$ introduced in Section 2 to the case of structured learning. A kernel function in this setting should reflect the dependencies between the instances and the labels, hence we define the structured kernel function as a function on the domain of the instances and the labels, namely, $k^S : (\mathcal{X} \times \mathcal{Y})^2 \to \mathbb{R}$. This kernel function induces the RKHS $\mathcal{H}^S$, where the inner product in this space is defined such that it satisfies the reproducing property, $\langle k^S((\mathbf{x}, y), \cdot), f\rangle = f(\mathbf{x}, y)$.

Initialize: $\mathcal{S}_0 = \emptyset$, $f_0 = \mathbf{0}$

For $t = 1, 2, \ldots$

Receive new instance $\mathbf{x}_t$

Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$

Receive label y_t

If $y_t \neq \hat{y}_t$ (*prediction error*)

Set $f'_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$

Set $f''_t = P_{t-1} f'_t$

Set $\delta_t = f''_t - f'_t$

If $\|\delta_t\| \leq \eta$

$f_t = f''_t$

$\mathcal{S}_t = \mathcal{S}_{t-1}$

Else

$f_t = f'_t$

$\mathcal{S}_t = \mathcal{S}_{t-1} \cup \mathbf{x}_t$

Else If $y_t = \hat{y}_t$ and $y_t f_{t-1}(\mathbf{x}_t) \leq 1$ (*margin error*)

Set $\delta_t = P_{t-1} k(\mathbf{x}_t, \cdot) - k(\mathbf{x}_t, \cdot)$

If $\ell_1(f_{t-1}(\mathbf{x}_t), y_t) \geq \frac{\|\delta_t\|}{\eta}$ (*check if the* $\mathbf{x}_t$ *can be projected onto* $\mathcal{H}_{t-1}$)

Set $\tau_t = \min \left\{ \frac{\ell_1(f_{t-1}(\mathbf{x}_t), y_t)}{\|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2}, 2\frac{\ell_1(f_{t-1}(\mathbf{x}_t), y_t) - \frac{\|\delta_t\|}{\eta}}{\|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2}, 1 \right\}$

Set $f_t = f_{t-1} + y_t \tau_t P_{t-1} k(\mathbf{x}_t, \cdot)$

$\mathcal{S}_t = \mathcal{S}_{t-1}$

Else

$f_t = f_{t-1}$

$\mathcal{S}_t = \mathcal{S}_{t-1}$

Else

$f_t = f_{t-1}$

$\mathcal{S}_t = \mathcal{S}_{t-1}$

Figure 5: The Projectron++ Algorithm.

As in the binary classification algorithm presented earlier, the structured output online algorithm receives instances in a sequential order. Upon receiving an instance, $\mathbf{x}_t \in \mathcal{X}$, the algorithm predicts a label, y'_t, according to Equation (16). After making its prediction, the algorithm receives the correct label, y_t. We define the loss suffered by the algorithm on round t for the example $(\mathbf{x}_t, y_t)$ as

$$\ell^S_\gamma(f, \mathbf{x}_t, y_t) = \max\{0, \gamma - f(\mathbf{x}_t, y_t) + \max_{y'_t \neq y_t} f(\mathbf{x}_t, y'_t)\},$$

and the cumulative loss D_γ^S as

$$D_\gamma^S = \sum_{t=1}^{T} \ell_\gamma^S(f, \mathbf{x}_t, y_t) \ .$$

Note that sometimes it is useful to define γ as a function $\gamma : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ describing the discrepancy between the predicted label and the true label. Our algorithm can handle such a label cost function, but we will not discuss this issue here (see Crammer et al., 2006, for further details).

As in the binary case, on rounds in which there is a prediction mistake, $y_t' \neq y_t$, the algorithm updates the hypothesis f_{t-1} by adding $k((\mathbf{x}_t, y_t), \cdot) - k((\mathbf{x}_t, y_t'), \cdot)$ or its projection. When there is a margin mistake, $0 < \ell_\gamma^S(f_{t-1}, \mathbf{x}_t, y_t) < \gamma$, the algorithm updates the hypothesis f_{t-1} by adding $\tau_t P_{t-1}\left(k((\mathbf{x}_t, y_t), \cdot) - k((\mathbf{x}_t, y_t'), \cdot)\right)$, where $0 < \tau_t < 1$ and will be defined shortly. Now, for the structured output case, δ_t is defined as

$$\delta_t = k((\mathbf{x}_t, y_t), \cdot) - k((\mathbf{x}_t, y_t'), \cdot) - P_{t-1}\left(k((\mathbf{x}_t, y_t), \cdot) - k((\mathbf{x}_t, y_t'), \cdot)\right) \ .$$

The analysis of the structured output Projectron++ algorithm is similar to that provided for the binary case. We can easily obtain the generalization of Lemma 2 and Theorem 4 as follows

Lemma 5 *Let $(\hat{\mathbf{x}}, \hat{y})$ be an example, with $\hat{\mathbf{x}} \in \mathcal{X}$ and $\hat{y} \in \mathcal{Y}$. Denote by $f(\cdot)$ an hypothesis in $\mathcal{H}^S$. Let $q(\cdot) \in \mathcal{H}^S$. Then the following holds for any $y' \in \mathcal{Y}$:*

$$\langle f, q \rangle \geq \gamma - \ell_\gamma^S(f, \hat{\mathbf{x}}, \hat{y}) - \|f\| \cdot \left\| q - \left(k((\hat{\mathbf{x}}, \hat{y}), \cdot) - k((\hat{\mathbf{x}}, y'), \cdot)\right) \right\| \ .$$

Moreover if $f(\cdot)$ can be written as $\sum_{i=1}^m \alpha_i k((\mathbf{x}_i, y_i), \cdot)$ with $\alpha_i \in \mathbb{R}$ and $\mathbf{x}_i \in \mathcal{X}, i = 1, \cdots, m$, and q is the projection of $k((\hat{\mathbf{x}}, \hat{y}), \cdot) - k((\hat{\mathbf{x}}, y'), \cdot)$ in the space spanned by $k((\mathbf{x}_i, y_i), \cdot), i = 1, \cdots, m$, we have that

$$\langle f, q \rangle = f(\hat{\mathbf{x}}, \hat{y}) - f(\hat{\mathbf{x}}, y') \ .$$

Theorem 6 *Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be a sequence of instance-label pairs where $\mathbf{x}_t \in \mathcal{X}$, $y_t \in \mathcal{Y}$, and $\|k((\mathbf{x}_t, y), \cdot)\| \leq 1/2$ for all t and $y \in \mathcal{Y}$. Assuming that Projectron++ is run with $\eta > 0$, the number of prediction mistakes it makes on the sequence is bounded by*

$$\frac{\|g\|^2}{(1 - \eta\|g\|)^2} + \frac{D_1^S}{1 - \eta\|g\|} + \frac{\|g\|}{1 - \eta\|g\|} \sqrt{max\left(0, \frac{D_1^S}{1 - \eta\|g\|} - B\right)}$$

where g is an arbitrary function in $\mathcal{H}^S$, such that $\|g\| < \frac{1}{\eta}$,

$$\begin{aligned}
a &= P_{t-1}\left(k((\mathbf{x}_t, y_t), \cdot) - k((\mathbf{x}_t, y_t'), \cdot)\right) \\
0 < \tau_t &< \min\left\{ 2\frac{\ell_1^S(f_{t-1}, \mathbf{x}_t, y_t) - \frac{\|\delta_t\|}{\eta}}{\|a\|^2}, 1 \right\} \\
B &= \sum_{\{t : 0 < \ell_1^S(f_{t-1}, \mathbf{x}_t, y_t) < 1\}} \tau_t \left(2\ell_1^S(f_{t-1}, \mathbf{x}_t, y_t) - \tau_t \|a\|^2 - 2\frac{\|\delta_t\|}{\eta} \right) > 0 \ .
\end{aligned}$$

As in Theorem 4 there is some freedom in the choice of τ_t, and again we set it to

$$\tau_t = \min \left\{ \frac{\ell_1^S(f_{t-1}, \mathbf{x}_t, y_t)}{\|a\|^2}, 2\frac{\ell_1^S(f_{t-1}, \mathbf{x}_t, y_t) - \frac{\|\delta_t\|}{\eta}}{\|a\|^2}, 1 \right\}.$$

In the multiclass decision problem case, the kernel $k((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2))$ is simplified to $\delta_{y_1 y_2} k(\mathbf{x}_1, \mathbf{x}_2)$, where $\delta_{y_1 y_2}$ is the Kronecker delta. This corresponds to the use of a different prototype for each class. This simplifies the projection step, in fact $k((\mathbf{x}_t, y_t), \cdot)$ can be projected only onto the functions in S_{t-1} belonging to y_t, the scalar product with the other functions being zero. So instead of storing a single matrix $\mathbf{K}_{t-1}^{-1}$, we need to store m matrices, where m is the number of classes, each one being the inverse matrix of the Gram matrix of the functions of one class. This results in improvements in both memory usage and computational cost of the algorithm. To see this suppose that we have m classes, each with n vectors in the support set. Storing a single matrix means having a space and time complexity of $O(m^2 n^2)$ (cf. Section 3), while in the second case the complexity is $O(mn^2)$. We use this method in the multiclass experiments presented in Section 7.

6. Bounding Other Online Algorithms

It is possible to apply the technique in the basis of the Projectron algorithm to any conservative online algorithm. A conservative online algorithm is an algorithm that updates its hypothesis only on rounds on which it makes a prediction error. By applying Lemma 2 to a conservative algorithm, we can construct a bounded version of it with worst case mistake bounds. As in the previous proofs, the idea is to use Lemma 2 to bound the scalar product of the competitor and the projected function. This yields an additional term which is subtracted from the margin γ of the competitor.

The technique presented here can be applied to other online kernel-based algorithms. As an example, we apply our technique to ALMA$_2$ (Gentile, 2001). Again we define two hypotheses: a temporary hypothesis f_t', which is the hypothesis of ALMA$_2$ after its update rule, and a projected hypothesis, which is the hypothesis f_t' projected on the set $\mathcal{H}_{t-1}$ as defined in Equation (2). Define the projection error δ_t as $\delta_t = f_t' - f_t''$. The modified ALMA$_2$ algorithm uses the projected hypothesis f_t'' whenever the projection error is smaller than a parameter η, otherwise it uses the temporary hypothesis f_t'. We can state the following bound

Theorem 7 *Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be a sequence of instance-label pairs where $\mathbf{x}_t \in \mathcal{X}$, $y_t \in \{-1, +1\}$, and $k(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Let α, B and $C \in \mathbb{R}^+$ satisfy the equation*

$$C^2 + 2(1-\alpha)BC = 1 \ .$$

Assume ALMA$_2$(α;B,C) projects every time the projection error $\|\delta_t\|$ is less than $\eta \geq 0$, then the number of prediction mistakes it makes on the sequence is bounded by

$$\frac{D_\gamma}{\gamma - \eta} + \frac{\rho^2}{2} + \sqrt{\frac{\rho^4}{4} + \frac{\rho^2}{\gamma - \eta} D_\gamma + \rho^2}$$

where $\gamma > \eta$, $\rho = \frac{1}{C^2(\gamma-\eta)^2}$, and g is an arbitrary function in $\mathcal{H}$, such that $\|g\| \leq 1$.

Proof The proof follows the original proof presented in Gentile (2001). Specifically, according to Lemma 2, one can replace the relation $y_t \langle g, k(\mathbf{x}_t, \cdot) \rangle \geq \gamma - \ell_\gamma(g(\mathbf{x}_t), y_t)$ with $y_t \langle g, P_{t-1} k(\mathbf{x}_t, \cdot) \rangle \geq$

Data Set	Samples	Features	Classes	Kernel	Parameters
a9a (Platt, 1999)	32561	123	2	Gaussian	0.04
ijcnn1 (Prokhorov, 2001)	49990	22	2	Gaussian	8
news20.binary (Keerthi et al., 2005)	19996	1355191	2	Linear	-
vehicle (Duarte and Hu, 2004)	78823	100	2	Gaussian	0.125
synthetic (Dekel et al., 2007)	10000	2	2	Gaussian	1
mnist (Lecun et al., 1998)	60000	780	10	Polynomial	7
usps (Hull, 1994)	7291	256	10	Polynomial	13
timit (subset) (Lemel et al., 1986)	~ 150000	351	39	Gaussian	80

Table 1: Data sets used in the experiments

$\gamma - \eta - \ell_\gamma(g(\mathbf{x}_t), y_t)$, and further substitute $\gamma - \eta$ for γ. ■

7. Experimental Results

In this section we present experimental results that demonstrate the effectiveness of the Projectron and the Projectron++ algorithms. We compare both algorithms to the Perceptron algorithm, the Forgetron algorithm (Dekel et al., 2007) and the Randomized Budget Perceptron (RBP) algorithm (Cesa-Bianchi et al., 2006). For Forgetron, we choose the state-of-the-art "self-tuned" variant, which outperforms all of its other variants. We used the PA-I variant of the Passive-Aggressive algorithm (Crammer et al., 2006) as a baseline algorithm, as it gives an upper bound on the classification performance of the Projectron++ algorithm. All the algorithms were implemented in MATLAB using the DOGMA library (Orabona, 2009).

We tested the algorithms on several standard machine learning data sets:[2] *a9a*, *ijcnn1*, *news20.binary*, *vehicle (combined)*, *usps*, *mnist*. We also used a synthetic dataset and the acoustic-phonetic dataset *timit*. The synthetic dataset was built in the same way as in Dekel et al. (2007). It is composed of 10000 samples taken from two separate bi-dimensional Gaussian distributions. The means of the positive and negative samples are $(1,1)$ and $(-1,-1)$, respectively, while the covariance matrices for both are diagonal matrices with $(0.2,2)$ as their diagonal. The labels are flipped with a probability of 0.1 to introduce noise. The list of the data sets, their characteristics and the kernels used, are given in Table 1. The parameters of the kernels were selected to have the best performance with the Perceptron and were used for all the other algorithms to result in a fair comparison. The C parameter of PA-I was set to 1, to give an update similar to Perceptron and Projectron. All the experiments were performed over five different permutations of the training set.

Experiments with one setting of η. In the first set of experiments we compared the online average number of mistakes and the support set size of all algorithms. Both Forgetron and RBP work by discarding vectors from the support set, if the size of the support set reaches the budget size, B. Hence for a fair comparison, we set η to some value and selected the budget sizes of Forgetron and RBP to be equal to the final size of the support set of Projectron. In particular, in Figure 6, we set $\eta = 0.1$ in Projectron and ended up with a support set of size 793, hence $B = 793$. In Figure 6(a) the average online error rate for all algorithms on the *a9a* data set is plotted. Note that Projectron closely tracks Perceptron. On the other hand Forgetron and RBP stop improving

2. Downloaded from `http://www.sie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

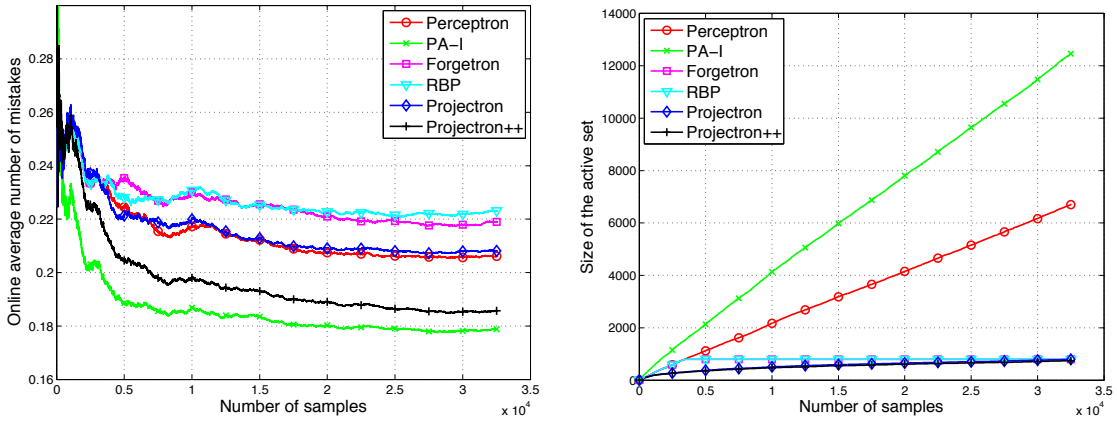


Figure 6: Average online error (left) and size of the support set (right) for the different algorithms on *a9a* data set as a function of the number of training samples (better viewed in color). B is set to 793, $\eta = 0.1$.

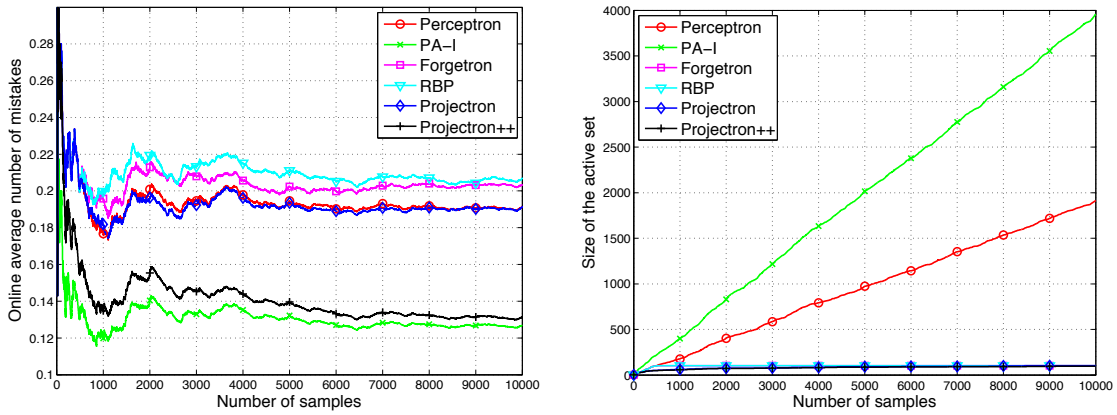
after reaching the support set size B, around 3400 samples. Moreover, as predicted by its theoretical analysis, Projectron++ achieves better results than Perceptron, even with fewer number of supports.

Figure 6(b) shows the growth of the support set as a function of the number of samples. While for the PA-I and the Perceptron the growth is clearly linear, it is sub-linear for Projectron and Projectron++: they will reach a maximum size and then they will stop growing, as stated in Theorem 1. Another important consideration is that Projectron++ outperforms Projectron *both* with respect to the size of the support set and number of mistakes. Using our MATLAB implementation, the running times for this experiment are $\sim$ 35s for RBP and Forgetron, $\sim$ 40s for Projectron and Projectron++, $\sim$ 130s for Perceptron, and $\sim$ 375s for PA-I. Hence Projectron and Projectron++ have a running time smaller than Perceptron and PA-I, due to their smaller support sets.

The same behavior can be seen in Figure 7, for the *synthetic* data set. Here the gain in performance of Projectron++ over Perceptron, Forgetron and RBP is even greater.

Experiments with a range of values for η - Binary. To analyze in more detail the behavior of our algorithms we decided to run other tests using a range of values of η. For each value we obtain a different size of the support set and a different number of mistakes. We used the data to plot a curve corresponding to the percentage of mistakes as a function of the support set size. The same curve was plotted for Forgetron and RBP, where the budget size was selected as described before. In this way we compared the algorithms along the continuous range of budget sizes, displaying the trade-off between sparseness and accuracy. For the remaining experiments we chose not to show the performance of Projectron, as it was always outperformed by Projectron++.

In Figure 8 we show the performance of the algorithms on different binary data sets: (a) *ijcnn1*, (b) *a9a*, (c) *news20.binary*, and (d) *vehicle (combined)*. Because Projectron++ used a different support set size for each permutation of the training samples, we plotted five curves, one for each of the five permutations. RBP and Forgetron have fixed budget sizes set in advance, hence for these algorithms we just plotted standard deviation bars, that are very small so they can be hardly seen in the figures. In all of the experiments Projectron++ outperforms Forgetron and RBP. One may



(a) (b)

Figure 7: Average online error (left) and size of the support set (right) for the different algorithms on the *synthetic* data set as a function of the number of training samples (better viewed in color). B is set to 103, $\eta = 0.04$.

note that there is a point in all the graphs where the performance of Projectron++ is better than Perceptron, and has a smaller support set. Projectron++ gets closer to the classification rate of the PA-I, without paying the price of a larger support set. Note that the performance of Projectron++ is consistently better than RBP and Forgetron, regardless of the kernel used, particularly, on the database *news20.binary*, which is a text classification task with linear kernel. In this task the samples are almost mutually orthogonal, so finding a suitable subspace on which to project is difficult. Nevertheless Projectron++ succeeded in obtaining better performance. The reason is probably due to the margin updates, which are performed without increasing the size of the solution. Note that a similar modification would not be trivial in Forgetron and in RBP, because the proofs of their mistake bounds strongly depend on the rate of growth of the norm of the solution.

Experiments with a range of values for η - Multiclass. We have also considered multiclass data sets, using the multiclass version of Projectron++. Due to the fact that there are no other bounded online algorithms with a mistake bound for multiclass, we have extended RBP in the natural manner to multiclass. In particular we used the *max-score* update in Crammer and Singer (2003), for which a mistake bound exists, discarding a vector at random from the solution each time a new instance is added and the number of support vectors is equal to the budget size. We name it Multiclass Random Budget Perceptron (MRBP). It should be possible to prove a mistake bound for this algorithm, extending the proof in Cesa-Bianchi et al. (2006). In Figure 9 we show the results for Perceptron, Passive-Aggressive, Projectron++ and MRBP trained on (a) *usps*, and (b) *mnist* data sets. The results confirm the findings found for the binary case.

The last data set used in our experiments is a corpus of continuous natural speech for the task of phoneme classification. The data we used is a subset of the TIMIT acoustic-phonetic data set, which is a phonetically transcribed corpus of high quality continuous speech spoken by North American speakers (Lemel et al., 1986). The features were generated from nine adjacent vectors of Mel-Frequency Cepstrum Coefficients (MFCC) along with their first and second derivatives. The TIMIT corpus is divided into a training set and a test set in such a way that no speakers from the training set

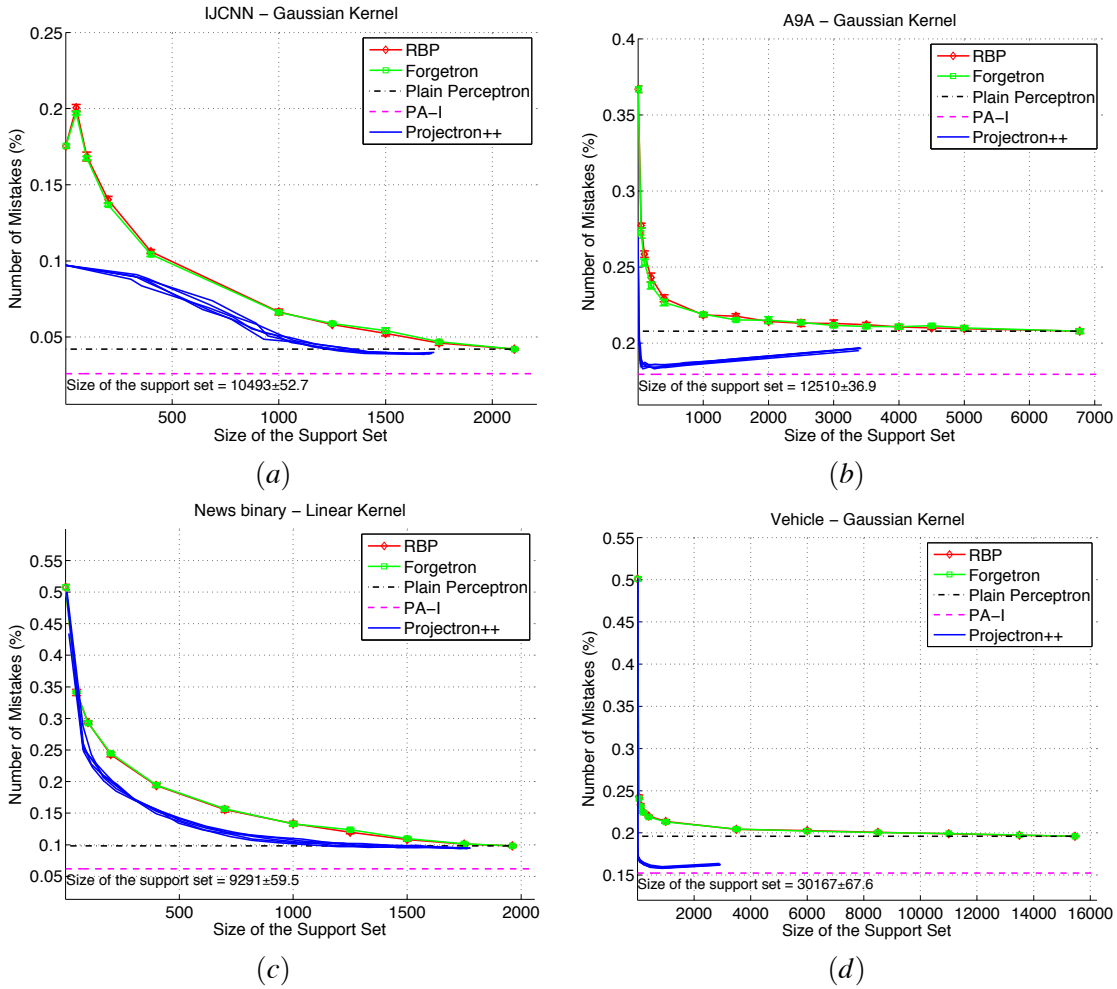
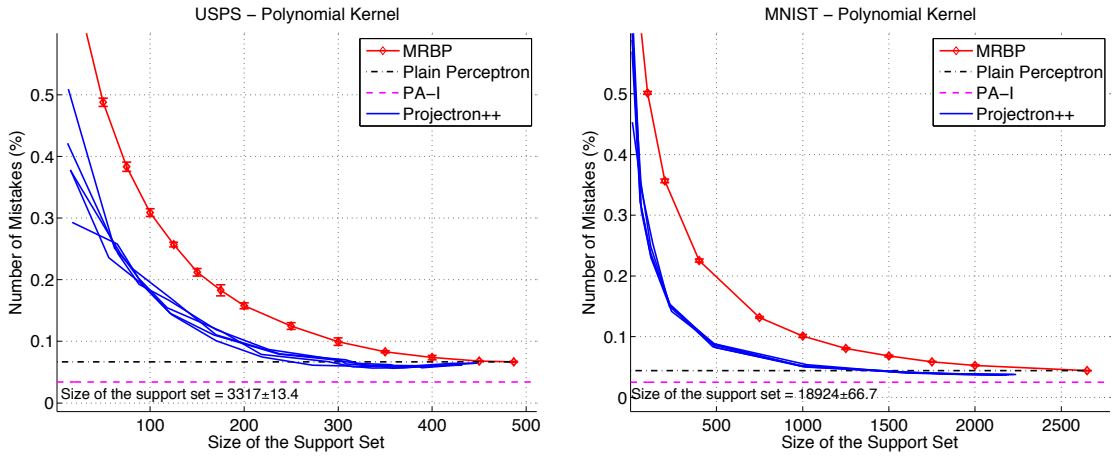


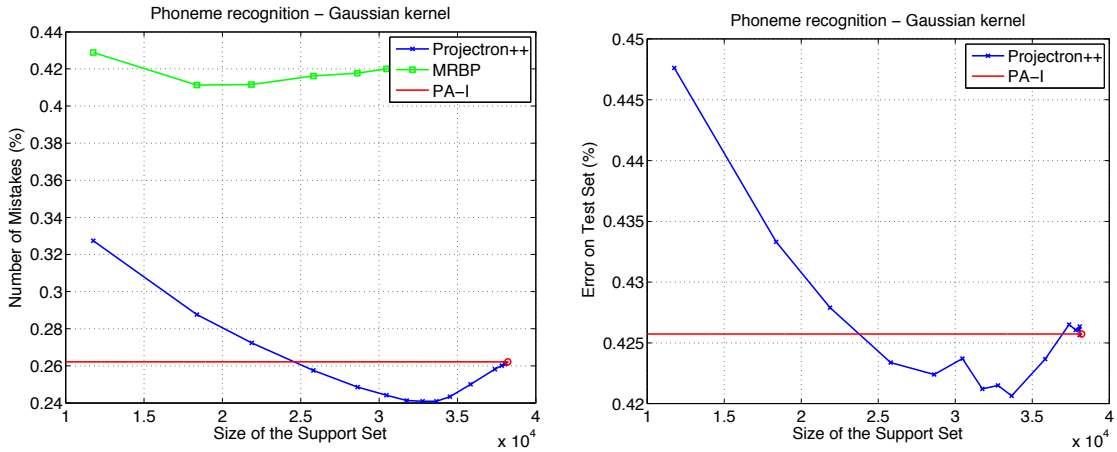
Figure 8: Average online error for the different algorithms as a function of the size of the support set on different binary data sets.

appear in the test set (speaker independent). We randomly selected 500 training utterances from the training set. The average online errors are shown in Figure 10 (a). We also tested the performance of the algorithm on the proposed TIMIT core test set composed of 192 utterances, the results of which are in Figure 10 (b). We used online-to-batch conversion (Cesa-Bianchi et al., 2004) to give a bounded batch solution. We did not test the performance of MRBP on the test set because for this algorithm the online-to-batch conversion does not produce a bounded solution. We compare the batch solution to the online-to-batch conversion of the PA-I solution. The results of Projectron++ are comparable to those of PA-I, while the former uses a smaller support set. These results also suggest that the batch solution is stable when varying the value of η, as the difference in performance on test set is less than 3%.



(a) (b)

Figure 9: Average online error for the different algorithms as a function of the size of the support set on different multiclass data sets.



(a) (b)

Figure 10: Average online error (a) and test error (b) for the different algorithms as a function of the size of the support set on a subset of the *timit* data set.

8. Discussion

This paper presented two different versions of a bounded online learning algorithm. The algorithms depend on a parameter that allows one to trade accuracy for sparseness of the solution. The size of the solution is always guaranteed to be bounded, although the size of this bound is unknown before the training begins. Therefore, these algorithms solve the memory explosion problem of the Perceptron and similar algorithms. Although the size of the support set cannot be determined

before training, practically, for a given target accuracy, the size of the support sets of Projectron or Projectron++ are much smaller than those of other budget algorithms such as Forgetron and RBP.

The first algorithm, Projectron, is based on the Perceptron algorithm. The empirical performance of Projectron is comparable to that of Perceptron, but with the advantage of a bounded solution. The second algorithm, Projectron++, introduces the notion of large margin and, for some values of η, outperforms the Perceptron algorithm, while assuring a bounded solution. The experimental results suggest that Projectron++ outperforms other online bounded algorithms such as Forgetron and RBP, with a similar hypothesis size.

There are two unique advantages of Projectron and Projectron++. First, these algorithms can be extended to the multiclass and the structured output settings. Second, a standard online-to-batch conversion can be applied to the online bounded solution of these algorithms, resulting in a bounded batch solution. The major drawback of these algorithms is their time and space complexity, which is quadratic in the size of the support set. Trying to overcome this acute problem is left for future work.

Acknowledgments

The authors would like to thank the anonymous reviewers for suggesting a way to improve Theorem 3. This work was supported by EU project DIRAC (FP6-0027787). The authors would like to thank Andy Cotter for proofreading the manuscript.

References

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems 14*, pages 409–415, 2000.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Trans. on Information Theory*, 50(9):2050–2057, 2004.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. *SIAM Journal on Computing*, 34(3):640–668, 2005.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. Tracking the best hyperplane with a simple budget Perceptron. In *Proc. of the 19th Conference on Learning Theory*, pages 483–498, 2006.

L. Cheng, S. V. N. Vishwanathan, D. Schuurmans, S. Wang, and T. Caelli. Implicit online learning with kernels. In *Advances in Neural Information Processing Systems 19*, pages 249–256, 2007.

M. Collins. Discriminative reranking for natural language parsing. In *Proc. of the 17th International Conference on Machine Learning*, pages 175–182, 2000.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

K. Crammer, J. Kandola, and Y. Singer. Online classification on a budget. In *Advances in Neural Information Processing Systems 16*, 2003.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14:641–668, 2002.

F. Cucker and D. X. Zhou. *Learning Theory: An Approximation Theory Viewpoint*. Cambridge University Press, New York, NY, USA, 2007.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372, 2007.

T. Downs, K. E. Gates, and A. Masters. Exact simplification of support vectors solutions. *Journal of Machine Learning Research*, 2:293–297, 2001.

M. F. Duarte and Y. H. Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64:826–838, 2004.

Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285, 2004.

Y. Freund and R. E. Schapire. Large margin classification using the Perceptron algorithm. *Machine Learning*, pages 277–296, 1999.

C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

C. Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3):265–299, 2003.

J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

S. S. Keerthi, D. Decoste, and T. Joachims. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6:2005, 2005.

J. Kivinen, A. Smola, and R. Williamson. Online learning with kernels. *IEEE Trans. on Signal Processing*, 52(8):2165–2176, 2004.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. In *Advances in Neural Information Processing Systems 21*, pages 905–912, 2008.

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.

L. Lemel, R. Kassel, and S. Seneff. Speech database development: Design and analysis. Report no. SAIC-86/1546, Proc. DARPA Speech Recognition Workshop, 1986.

F. Orabona. *DOGMA: A MATLAB Toolbox for Online Learning*, 2009. Software available at `http://dogma.sourceforge.net`.

F. Orabona, C. Castellini, B. Caputo, J. Luo, and G. Sandini. Indoor place recognition using online independent support vector machines. In *Proc. of the British Machine Vision Conference 2007*, pages 1090–1099, 2007.

J. C. 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*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

D. Prokhorov. IJCNN 2001 neural network competition. Technical report, 2001.

F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Proc. of the 14th Conference on Computational Learning Theory*, pages 416–426, London, UK, 2001. Springer-Verlag.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 17*, 2003.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. of the 21st International Conference on Machine Learning*, pages 823–830, 2004.

J. Weston, A. Bordes, and L. Bottou. Online (and offline) on an even tighter budget. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proc. of AISTATS 2005*, pages 413–420, 2005.

Journal of Machine Learning Research 10 (2009) 2667-2714 Submitted 2/07; Revised 3/09; Published 11/09

Structure Spaces

Brijnesh J. Jain JBJ@CS.TU-BERLIN.DE
Klaus Obermayer OBY@CS.TU-BERLIN.DE
Department of Electrical Engineering and Computer Science
Berlin University of Technology
Berlin, Germany

Editor: Tommi Jaakkola

Abstract

Finite structures such as point patterns, strings, trees, and graphs occur as "natural" representations of structured data in different application areas of machine learning. We develop the theory of *structure spaces* and derive geometrical and analytical concepts such as the angle between structures and the derivative of functions on structures. In particular, we show that the gradient of a differentiable structural function is a well-defined structure pointing in the direction of steepest ascent. Exploiting the properties of structure spaces, it will turn out that a number of problems in structural pattern recognition such as central clustering or learning in structured output spaces can be formulated as optimization problems with cost functions that are locally Lipschitz. Hence, methods from nonsmooth analysis are applicable to optimize those cost functions.

Keywords: graphs, graph matching, learning in structured domains, nonsmooth optimization

1. Introduction

In pattern recognition and machine learning, it is common practice to represent data by feature vectors living in a Banach space, because this space provides powerful analytical techniques for data analysis, which are usually not available for other representations. A standard technique to solve a learning problem in a Banach space is to set up a smooth error function, which is then minimized by using local gradient information.

But often, the data we want to learn about have no natural representation as feature vectors and are more naturally represented in terms of finite combinatorial structures such as, for example, point patterns, strings, trees, lattices or graphs. Such learning problems arise in a variety of applications, which range from predicting the biological activity of a given chemical structure over finding frequent substructures of a data set of chemical compounds, and predicting the 3D-fold of a protein given its amino sequence, to natural language parsing, to name just a few.

In many applications, the set $\mathcal{X}$ of finite combinatorial structures is equipped with a distance function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$, which is often provided by external knowledge. An example of such a distance function is the edit distance on string, trees, or graphs (Levenshtein, 1966; Sanfeliu and Fu, 1983; Shapiro and Haralick, 1985; Shasha and Zhang, 1989; Zhang, 1996). The edit distance is applied to sequence alignment in bioinformatics (Gusfield, 1997), in chemoinformatics (Raymond and Willett, 2002) by means of the maximum common subgraph isomorphism, and in computer vision (Eshera and Fu, 1986; Myers, Wilson, and Hancock, 2000; Robles-Kelly and Hancock, 2005). Since

distance spaces $(\mathcal{X},d)$ of structures often have less mathematical structure than Banach spaces, several standard statistical pattern recognition techniques cannot be easily applied to $(\mathcal{X},d)$.

There are two main approaches that apply standard statistical pattern recognition techniques to a given distance space $(\mathcal{X},d)$. The first approach directly operates on the space $(\mathcal{X},d)$. Examples are the k-nearest neighbor classifier, the linear programming machine (Graepel, Herbrich, Bollmann-Sdorra, and Obermayer, 1999), and pairwise clustering (Hofmann and Buhmann, 1997; Graepel and Obermayer, 1999). These methods can cope with sets $\mathcal{X}$ that possess an arbitrary distance function d as the sole mathematical structure on $\mathcal{X}$. The problem is that many pattern recognition methods require a space $\mathcal{X}$ with a richer mathematical structure. For example, large margin classifiers require as mathematical structure a complete vector space in which distances and angles can be measured. From an algorithmic point of view, many pattern recognition methods use local gradient information to minimize some cost function. For these methods, Banach spaces are endowed with enough structure to define derivatives and gradients.

The aim of the second approach is to overcome the lack of mathematical structure by embedding a given distance space $(\mathcal{X},d)$ into a mathematically richer space $(\mathcal{X}',d')$. Several methods have been proposed, which mainly differ in the choice of the target space $\mathcal{X}'$ and to which extent the original distance function d is preserved. Typical examples are embeddings into Euclidean spaces (Cox and Cox, 2000; Luo, Wilson, and Hancock, 2003; Minh and Hofmann, 2004), Hilbert spaces (Gärtner, 2003; Hochreiter and Obermayer, 2004, 2006; Kashima, Tsuda, and Inokuchi, 2003; Lodhi, Saunders, Shawe-Taylor, Cristianini, and Watkins, 2002), Banach spaces (Hein, Bousquet, and Schölkopf, 2005; von Luxburg and Bousquet, 2004), and Pseudo-Euclidean spaces (Herbrich, Graepel, Bollmann-Sdorra, and Obermayer, 1998; Goldfarb, 1985; Pekalska, Paclik, and Duin, 2001).

During this transformation, one has to ensure that the relevant information of the original problem is preserved. Under the assumption that d is a reasonable distance function on $\mathcal{X}$ provided by some external knowledge, we can preserve the relevant information by isometrically embedding the original space $(\mathcal{X},d)$ into some target space $(\mathcal{X}',d')$. Depending on the choice of the target space this is only possible if the distance function d satisfies certain properties. Suppose that $\mathcal{S}=\{x_1,\ldots,x_k\}\subseteq\mathcal{X}$ is a finite set and $D=(d_{ij})$ is a distance matrix with elements $d_{ij}=d(x_i,x_j)$. If d is symmetric and homogeneous, we can isometrically embed $\mathcal{D}$ into a Pseudo-Euclidean space (Goldfarb, 1985). In the case that d is a metric, the elements of $\mathcal{D}$ can be isometrically embedded into a Banach space. An isometric embedding of $\mathcal{S}$ into a Hilbert or Euclidean space is possible only if the matrix D^2 is of negative type (Schoenberg, 1937).[1]

Most standard learning methods have been developed in a Hilbert space or in a Euclidean space equipped with a Euclidean distance. But distance matrices of a finite set of combinatorial structures are often not of negative type and therefore an isometric embedding into a Hilbert space or Euclidean space is not possible. Another common problem of most isometric embeddings is that they only preserve distance relations and disregard knowledge about the inherent nature of the elements from the original space. For example the inherent nature of graphs is that they consist of a finite set of vertices together with a binary relation on that set. These information is lost, once we have settled in the target space for solving a pattern recognition problem. But for some methods in pattern recognition it is necessary to either directly access the original data or to recover the effects of the operations performed in the target space. One example is the sample mean of a set of combinatorial

1. A symmetric matrix M is of negative type if $x^TMx <= 0$ for all x with $x^T1=0$.

structures (Jain and Obermayer, 2008; Jiang, Münger, and Bunke, 2001), which is a fundamental concept for several methods in pattern recognition such as principal component analysis and central clustering (Gold, Rangarajan, and Mjolsness, 1996; Günter and Bunke, 2002; Lozano and Escolano, 2003; Jain and Wysotzki, 2004; Bonev, Escolano, Lozano, Suau, Cazorla, and Aguilar, 2007; Jain and Obermayer, 2008). The sample mean of a set of vectors is the vector of sample means of each component of those vectors. Similarly, a sample mean of a set of combinatorial structures is a combinatorial structure composed of the sample means of the constituents parts the structure is composed of. Another example is finding frequent substructures in a given set of combinatorial structures (Dehaspe, Toivonen, and King, 1998; Yan and Han, 2002). For such problems a principled framework is missing.

In this contribution, we present a theoretical framework that isometrically and isostructurally embeds certain metric spaces $(\mathcal{X},d)$ of combinatorial structures into a quotient space $(\mathcal{X}',d')$ of a Euclidean vector space. Instead of discarding information about the inherent nature of the original data, we can weaken the requirement that the embedding of $(\mathcal{X},d)$ into $(\mathcal{X}',d')$ should be isometric for all metrics. Here, we focus on metrics d that are related to the pointwise maximum of a set of Euclidean distances. This restriction is acceptable from an application point of view, because we can show that such metrics on combinatorial structures and their related similarity functions are a common choice of proximity measure in a number of different applications (Gold, Rangarajan, and Mjolsness, 1996; Holm and Sander, 1993; Caprara, Carr, Istrail, Lancia, and Walenz, 2004).

The quotient space $(\mathcal{X}',d')$ preserves the distance relations and the nature of the original data. The related Euclidean space provides the mathematical structure that gives rise to a rich arsenal of learning methods. The goal of the proposed approach is to adopt standard learning methods based on local gradient information to learning on structures in the quotient space $\mathcal{X}'$. In order to do so, we need an approach that allows us to formally adopt geometrical and analytical concepts for finite combinatorial structures. The proposed approach maps combinatorial structures to equivalence classes of vectors, where the elements of the same equivalence class are different vector representations of the same structure. Mapping a combinatorial structure to an equivalence class of vectors rather than to a single vector provides a link to the geometry of Euclidean spaces and at the same time preserves the nature of the original data. The resulting quotient set (the set of equivalence classes) leads to the more abstract notion of $\mathcal{T}$-space. Formally, a $\mathcal{T}$-space $\mathcal{X}_{\mathcal{T}}$ over a vector space $\mathcal{X}$ is a quotient set of $\mathcal{X}$, where the equivalence classes are the orbits of the group action of a transformation group $\mathcal{T}$ on $\mathcal{X}$. We show that $\mathcal{T}$-spaces encompass a variety of different classes of combinatorial structures, which also includes vectors. Thus, the theory of $\mathcal{T}$-spaces generalizes the vector space concept to cope with combinatorial structures and aims at retaining the geometrical and algebraic properties of a vector space to a certain extent.

We present case studies to illustrate that the theoretical framework can be applied to machine learning applications.

This paper is organized as follows: Section 2 provides an overview about the basic idea of the proposed approach. In Section 3, we study $\mathcal{T}$-spaces $\mathcal{X}_{\mathcal{T}}$ over metric, normed, and inner product vector spaces $\mathcal{X}$. We show that the gradient of a smooth function on structures satisfies the necessary condition of optimality and is a well-defined structure pointing in direction of steepest ascent. In Section 4, we use the theory of $\mathcal{T}$-spaces to formulate selected problems in structural pattern recognition as continuous optimization problems. We show that the proposed cost functions are locally Lipschitz and therefore nonsmooth on a set of Lebesgue measure zero. For this class of functions, we can apply methods from nonsmooth optimization. As a case study, we discuss in Section 5 the

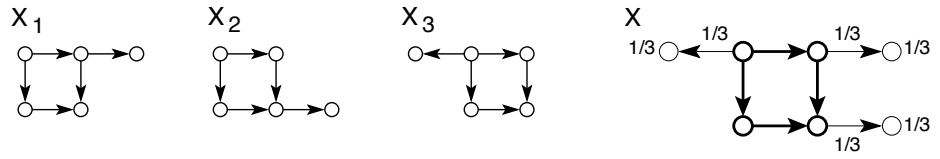


Figure 1: Illustration of a sample mean $\overline{X}$ of the three graphs $\mathcal{D} = \{X_1, X_2, X_3\}$. Vertices and edges of $\overline{X}$ that occur in all of the three example graphs from $\mathcal{D}$ are highlighted with bold lines. All other vertices and edges of $\overline{X}$ are annotated with the relative frequency of their occurrence in $\mathcal{D}$. By annotating the highlighted vertices and edges of $\overline{X}$ with 1, we obtain a weighted graph.

problem of determining a sample mean of a set of structures including its application to central clustering. As structures we consider point patterns and attributed graphs. Section 6 concludes. Technical parts and proofs have been delegated to the appendix.

2. An Example

The purpose of this section is to provide an overview about the basic idea of the proposed approach. To this end, we consider the (open) problem of determining the sample mean of graphs as a simple introductory example. The concept of a sample mean is the theoretical foundation for central clustering algorithms (see Section 4.3 and references therein).

A *directed graph* is a pair $X = (V, E)$ consisting of a finite set V of *vertices* and a set $E = \{(i, j) \in V \times V : i \neq j\}$ of *edges*.

By $\mathcal{G}$ we denote the set of all directed graphs. Suppose that

$$\mathcal{D} = (X_1, \ldots, X_k)$$

is a collection of k not necessarily distinct graphs from $\mathcal{G}$. Our goal is to determine a sample mean of $\mathcal{X}$. Intuitively, a sample mean averages the occurrences of vertices and edges within their structural context as illustrated in Figure 1.

As the sample mean of integers is not necessarily an integer, the sample mean of $\mathcal{D}$ is not necessarily a directed graph from $\mathcal{G}$ (see Figure 1). Therefore, we extend the set $\mathcal{G}$ of directed graphs to the set $\mathcal{G}[\mathbb{R}]$ of weighted directed graphs. A *weighted directed graph* is a triple $X = (V, E, \alpha)$ consisting of a directed graph (V, E) and a *weight function* $\alpha : V \cup V \to \mathbb{R}$ such that each edge has nonzero weight. A weighted directed graph X of order $|V| = n$ is completely specified by its *weight matrix* $X = (x_{ij})$ with elements $x_{ij} = \alpha(i, j)$ for all $i, j \in \{1, \ldots, n\}$.

The standard method

$$\overline{X} = \frac{1}{k} \sum_{i=1}^{k} X_i$$

to determine the sample mean $\overline{X}$ of $\mathcal{D}$ fails, because a well-defined addition of directed graphs is unknown as indicated by Figure 2. Therefore, we consider an equivalent characterization of the standard notion of sample mean. Following Jiang, Münger, and Bunke (2001), we adopt the optimization formulation of the standard sample mean. For vectors, the sample mean minimizes the

sum of squared Euclidean distances from the data points. In line with this formulation, we define a sample mean of $\mathcal{D}$ as a global minimum of the cost function

$$F(X) = \sum_{i=1}^{k} D(X, X_i)^2, \tag{1}$$

where D is some appropriate distance function on $\mathcal{G}[\mathbb{R}]$ that measures structural consistent and inconsistent parts of the graphs under consideration.

In principle, we could use any "well-behaved" distance function.[2] Here, we first consider distance functions on structures that generalize the Euclidean metric, because Euclidean spaces have a rich repository of analytical tools. To adapt at least parts of these tools for structure spaces, it seems to be reasonable to relate the distance function D in Equation (1) to the Euclidean metric. From an application point of view, this restriction is acceptable for the following reasons: (i) Geometric distance functions on graphs and their related similarity functions are a common choice of proximity measure in a number of different applications (Gold, Rangarajan, and Mjolsness, 1996; Holm and Sander, 1993); and (ii) it can be shown that a number of structure-based proximity measures like, for example, the maximum common subgraph (Raymond and Willett, 2002) or maximum contact map overlap problem for protein structure comparison (Goldman, Istrail, and Papadimitriou, 1999) can be related to an inner product and therefore to the Euclidean distance.

The geometric distance functions D we consider here are usually defined as the maximum of a set of Euclidean distances. This definition implies that (i) the cost function F is neither differentiable nor convex; (ii) the sample mean of graphs is not unique as shown in Figure 2; and (iii) determining a sample mean of graphs is NP-complete, because evaluation of D is NP-complete. Thus, we are faced with an intractable combinatorial optimization problem, where, at a first glance, a solution has to be found from an uncountable infinite set. In addition, multiple local minima of the cost function F complicates a characterization of a structural mean.

To deal with these difficulties, we embed graphs into a $\mathcal{T}$-space as we will show shortly. The basic idea is to view graphs as equivalence classes of vectors via their weight matrices, where the elements of the same equivalence class are different vector representations of the same structure. The resulting quotient set (the set of equivalence classes) leads to the more abstract notion of $\mathcal{T}$-space. Formally, a $\mathcal{T}$-space $\mathcal{X}_{\mathcal{T}}$ over a vector space $\mathcal{X}$ is a quotient set of $\mathcal{X}$, where the equivalence classes are the orbits of the group action of a transformation group $\mathcal{T}$ on $\mathcal{X}$. The theory of $\mathcal{T}$-spaces generalizes the vector space concept to cope with combinatorial structures and aims at retaining the geometrical and algebraic properties of a vector space to a certain extent. In doing so, the $\mathcal{T}$-space concept not only clears the way to approach the structural version of the sample mean in a principled way, but also generalizes standard techniques of learning in structured domains.

2.1 The Basic Approach

To construct $\mathcal{T}$-spaces, we demand that all graphs are of bounded order n, where the bound n can be chosen arbitrarily large. For a pattern recognition application this is not a serious restriction, because we can assume that the data graphs of interest are of bounded order. In the second step, we align each weighted directed graph X of order $m < n$ to a graph X' of order n by adding $p = n - m$

2. As we will see later, a distance function is well-behaved if it is locally Lipschitz.

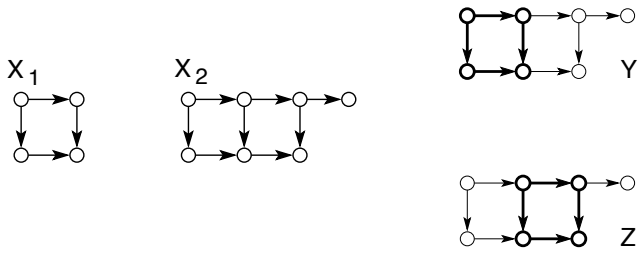


Figure 2: Illustration of one key problem in the domain of graphs: the lack of a well-defined addition. The graph X_1 can be added to X_2 with respect to $D(X_1, X_2)$ in two different ways as indicated by the highlighted subgraphs of Y and Z. As a consequence, Y and Z can be regarded as two distinct sample means of X_1 and X_2.

isolated vertices. The weighted adjacency matrix of the aligned graph X' is then of the form

$$X' = \begin{pmatrix} X & 0_{m,p} \\ 0_{p,m} & 0_{p,p} \end{pmatrix},$$

where X is the weighted adjacency matrix of X, and $0_{m,p}, 0_{p,m}, 0_{p,p}$ are padding zero matrices. By $\mathcal{G}[\mathbb{R}, n]$ we denote the set of weighted directed graphs of bounded order n.

For practical issues, it is important to note that restricting to structures of bounded order n and alignment of structures are purely technical assumptions to simplify mathematics. For machine learning problems, these limitations should have no practical impact, because neither the bound n needs to be specified nor an alignment of all graphs to an identical order needs to be performed. In a practical setting, we cancel out both technical assumptions by considering structure preserving mappings between the vertices of X and Y. Thus, when applying the theory, all we actually require is that the graphs are finite. We will return to this issue later, when we have provided the necessary technicalities.

The positions of the diagonal elements of X determine an ordering of the vertices. Conversely, different orderings of the vertices may result in different matrices. Since we are interested in the structure of a graph, the ordering of its vertices does not really matter. Therefore, we consider two matrices X and X' as being equivalent, denoted by $X \sim X'$, if they can be obtained from one another by reordering the vertices. Mathematically, the equivalence relation can be written as

$$X \sim X' \Leftrightarrow \exists P \in \mathcal{T} : P^{\mathsf{T}} X P = X',$$

where $\mathcal{T}$ denotes the set of all $(n \times n)$-permutation matrices.[3] The set $\mathcal{T}$ together with the function composition $T \circ T'$ for all $T, T' \in \mathcal{T}$ forms an algebraic group. By $[X]$ we denote the equivalence class of all matrices equivalent to X. Occasionally, we also refer to $[X]$ as the equivalence class of the graph X.

There are $n!$ different orderings of the vertices for an arbitrary graph X with n vertices. Each of the $n!$ orderings determines a weighted adjacency matrix. The equivalence class of X consist of all its different matrix representation. Note that different orderings of the vertices may result in the same matrix representation of the graph.

3. The letter $\mathcal{T}$ stands for transformation.

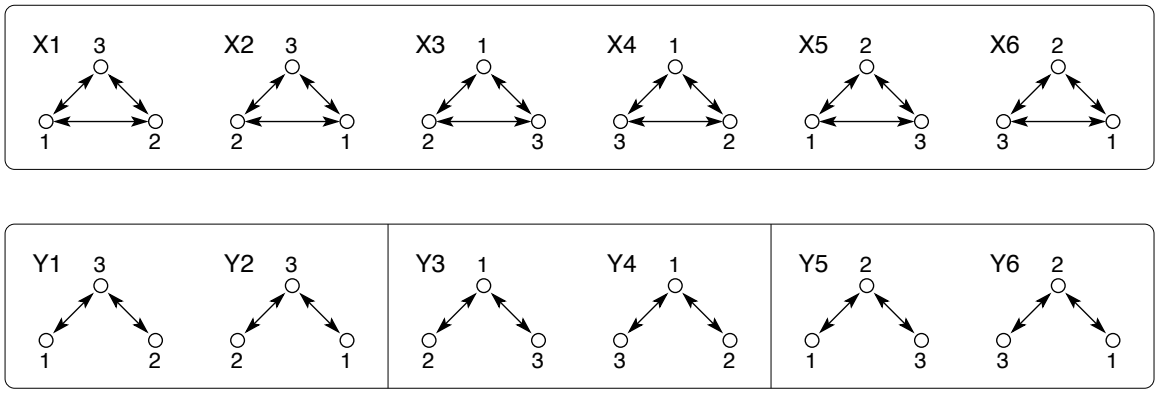


Figure 3: Illustration of two graphs with all possible orderings of their vertices. The number attached to the vertices represents their order (and are *not* attributes). The ordered graphs are grouped together according to their matrix representations. All ordered graphs $X1$-$X6$ yield the same weighted adjacency matrix. In the second row, the pairs $(Y1, Y2)$, $(Y3, Y4)$, and $(Y5, Y6)$ result in identical matrices.

Example 1 *Consider the graphs $X = X1$ and $Y = Y1$ depicted in the first column of Figure 3. The numbers annotated to the vertices represent an arbitrarily chosen ordering. Suppose that all vertices and edges have attribute* 1*. Then the weighted adjacency matrices of X and Y given the chosen ordering of their vertices are of the form*

$$X = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad \text{and} \qquad Y = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

The first and second row of Figure 3 show the $3! = 6$ different orderings of X and Y, respectively.

The matrix representation of X is independent of its ordering, that is, each reordering of the vertices of X results in the same matrix representation. Hence, the equivalence class of X consists of the singleton X.

The 6 *different orderings of graph Y result in three different matrix representations. The equivalence class of Y is of the form*

$$[Y] = \left\{ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \right\},$$

where the first matrix refers to the ordering of $Y = Y1$ and $Y2$, the second to $Y3$ and $Y4$, and the third to $Y5$ and $Y6$.

Since we may regard matrices as vectors, we can embed X into the vector space $\mathcal{X} = \mathbb{R}^{n \times n}$ as the set $[X]$ of all vector representations of X. We call the quotient set

$$\mathcal{X}_{\mathcal{T}} = \mathcal{X} / \sim = \bigcup_{X \in \mathcal{X}} [X]$$

consisting of all equivalence classes *$\mathcal{T}$-space* over the *representation space* $\mathcal{X}$. Figure 4 depicts an embedding of a graph into a vector space.

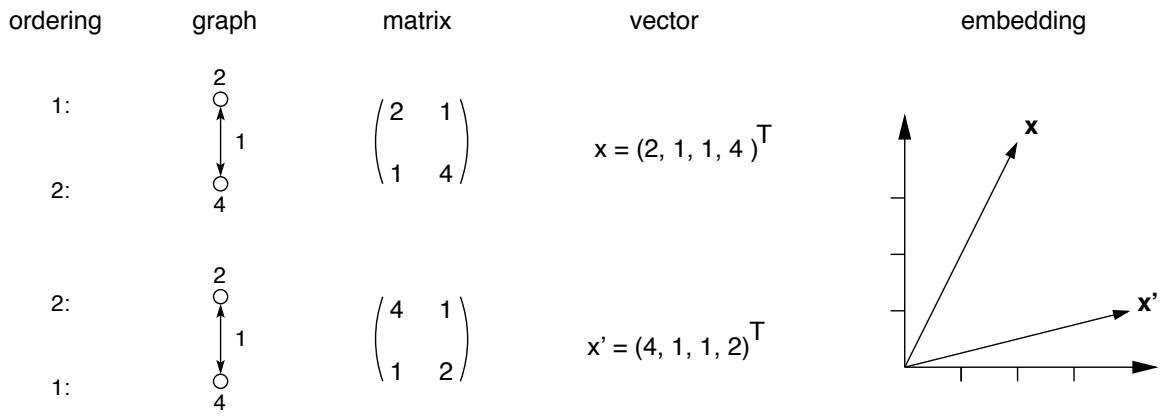


Figure 4: Illustration of an embedding of a graph of order 2. The attributes of the vertices are 2 and 4, and the attribute of the bidirectional edge is 1. Depending on the ordering of the vertices, we obtain two different matrix representations. Stacking the columns of the matrix to a 4-dimensional vector yields the vector representations x and x'. The plot in the last column depicts both representation vectors by considering their first and fourth dimension. Thus a graph is represented as a set of vectors in some vector space.

3. $\mathcal{T}$-spaces

In this section, we formalize the ideas on $\mathcal{T}$-spaces of the previous section. We consider a more general setting in the sense that we include classes of finite structures other than directed graphs with attributes from arbitrary vector spaces rather than weights from $\mathbb{R}$. The chosen approach that allows to formally adopt geometrical and analytical concepts makes use of the notion of *r-structures*. We introduce r-structures in Section 3.1. Based on the notion of r-structures, we develop the theory of $\mathcal{T}$-spaces in Sections 3.2 and 3.3. For a detailed technical treatment of $\mathcal{T}$-spaces we refer to Appendix A and B. Finally, Section 3.4 considers optimization of locally Lipschitz functions on $\mathcal{T}$spaces.

3.1 Attributed r-Structures

A *r-structure* is a pair $X = (\mathcal{P}, \mathcal{R})$ consisting of a finite set $\mathcal{P} \neq \emptyset$, and a subset $\mathcal{R} \subseteq \mathcal{P}^r$. The elements of $\mathcal{P}$ are the *points* of the r-structure X, the elements of $\mathcal{R}$ are its r-ary *relations*. A r-structure with points $\mathcal{P}$ is said to be a r-structure *on* $\mathcal{P}$. For convenience, we occasionally identify the structure X on $\mathcal{P}$ with its relation $\mathcal{R}$.

The following examples serve to indicate that several types of combinatorial structures can be regarded as r-structures. We first show that graphs are 2-structures. For this, we use the following notation: Given a finite set $\mathcal{P}$ of points, let

$$\mathcal{P}^{[2]} = \{(p,q) : p,q \in \mathcal{P}, p \neq q\}$$

be the set of tuples from $\mathcal{P}^2$ without diagonal elements (p,p).

Example 2 (Graphs) *Let $\mathcal{P}$ be a finite set of points, and let $X = (\mathcal{P}, \mathcal{R})$ be a 2-structure with $\mathcal{R} \subseteq \mathcal{P}^2$.*

1. *X is a* directed graph *if* $\mathcal{R} \subseteq \mathcal{P}^{[2]}$.

2. *X is a* simple graph *if* $\mathcal{R} \subseteq \mathcal{P}^{[2]}$ *such that* $(p,q) \in \mathcal{R}$ *implies* $(q,p) \in \mathcal{R}$.

3. *X is a* simple graph with loops *if* $\mathcal{R} \subseteq \mathcal{P}^2$ *such that* $(p,q) \in \mathcal{R}$ *implies* $(q,p) \in \mathcal{R}$. *Loops are edges* (p,p) *with the same endpoints*.

In a similar way, we can define further types of graphs such as, for example, trees, directed acyclic graphs, complete graphs, and regular graphs as 2-structures by specifying the corresponding properties on $\mathcal{R}$.

The next example shows that elements of a set are 1-structures.

Example 3 (Set of Elements) *Let $\mathcal{P}$ be a finite set of points. The* elements $E(\mathcal{P}) = (\mathcal{P}, \mathcal{R})$ *is a* 1*-structure with $\mathcal{R} = \mathcal{P}$, that is its relations are the elements of $\mathcal{P}$.*

To introduce analytical concepts to functions on r-structures, we shift from discrete to continuous spaces by introducing attributes. Let $\mathcal{A} = \mathcal{R}^d$ denote the set of attributes. An *$\mathcal{A}$-attributed r-structure* is a triple $X_\alpha = (\mathcal{P}, \mathcal{R}, \alpha)$ consisting of a r-structure $X = (\mathcal{P}, \mathcal{R})$ and an *attribution* $\alpha : \mathcal{P}^r \to \mathcal{A}$ with $\alpha(p) \neq 0$ if, and only if, $p \in \mathcal{R}$. Besides the technical argument, attributions also have a practical relevance, because they are often used to enhance descriptions of structured objects. The next example collects some attributed structures.

Example 4 *Let $\mathcal{P}$ be a finite set of order n.*

- Attributed graphs*: Let $\mathcal{A} = \mathbb{R}^d$. An* attributed graph *is an $\mathcal{A}$-attributed* 2*-structure $G_\alpha = (\mathcal{P}, \mathcal{R}, \alpha)$, where $G = (\mathcal{P}, \mathcal{R})$ is a simple graph with loops and $\alpha : \mathcal{R} \to \mathcal{A}$ is an attribution that assigns each vertex (loop) and each edge a non-zero feature vector.*

- Point patterns*: Let $\mathcal{A} = \mathbb{R}^2$. A* point pattern *is an $\mathcal{A}$-attributed* 1*-structure $P_\alpha = (\mathcal{P}, \mathcal{R}, \alpha)$, where $E(\mathcal{P}) = (\mathcal{P}, \mathcal{R})$ are the elements of $\mathcal{P}$ and $\alpha : \mathcal{R} \to \mathcal{A}$ is an attribution that assigns each element $p \in \mathcal{P}$ its coordinates $\alpha(p)$.*

The next example shows that vectors are attributed 1-structures. Hence, all results on r-structures are also valid in vector spaces.

Example 5 *Let $\mathcal{A}$ be a vector space. Suppose that $\mathcal{P}$ is of order $n = 1$. A* vector *is an $\mathcal{A}$-attributed* 1*-structure $x_\alpha = (\mathcal{P}, \mathcal{R}, \alpha)$, where $E(\mathcal{P}) = (\mathcal{P}, \mathcal{R})$ is the single element of $\mathcal{P}$ and $\alpha : \mathcal{R} \to \mathcal{A}$ is an attribution that maps a singleton to a vector. Hence, the set of all possible structures x_α on $\mathcal{P}$ reproduces the vector space $\mathcal{A}$.*

Note that we may assume without loss of generality that the attributes of r-relations from $\mathcal{R}$ are nonzero, that is $\alpha(\mathcal{R}) \subseteq \mathcal{A} \setminus \{0\}$. If the zero vector 0 is required as a valid attribute of a r-relation, we can always change, for example, to the vector space $\mathcal{A}' = \mathcal{A} \times \mathbb{R}$ and redefine α as

$$\alpha' : \mathcal{P}^r \to \mathcal{A} \times \mathbb{R}, \quad p \mapsto \begin{cases} (0,0) & : \quad p \in \mathcal{P}^r \setminus \mathcal{R} \\ (\alpha(p), 1) & : \quad p \in \mathcal{R} \end{cases}.$$

An $\mathcal{A}$-attributed r-structure $X = (\mathcal{P}, \mathcal{R}, \alpha)$ is completely specified by its *matrix representation* $X = (x_{p_1 \ldots p_r})$ with elements

$$x_{p_1 \ldots p_r} = \alpha(p_1, \ldots, p_r)$$

for all $p = (p_1, \ldots, p_r) \in \mathcal{P}^r$. For example, the matrix representation of a simple graph is its ordinary adjacency matrix.

Neither the "nature" of the points $\mathcal{P}$ nor the particular form of the r-relations $\mathcal{R}$ of a given r-structure $X = (\mathcal{P}, \mathcal{R}, \alpha)$ do really matter. What matters is the structure described by $\mathcal{R}$. Suppose that X is of order $|\mathcal{P}| = n$. To abstract from the "nature" of points, we choose $\mathbb{Z}_n = \{1, \ldots, n\}$ as our standard set of points. The particular form of $\mathcal{R}$ depends on the numbering of the points from $\mathbb{Z}_n$. To abstract from the particular form of $\mathcal{R}$, we identify sets of r-relations that can be obtained from one another by renumbering the points. Mathematically, we can express these sets by means of *isomorphism classes*. Two $\mathcal{A}$-attributed r-structures $X = (\mathcal{P}, \mathcal{R}, \alpha)$ and $X' = (\mathcal{P}', \mathcal{R}', \alpha')$ are *isomorphic*, written as $X \simeq X'$ if there is a bijective mapping $\phi : \mathcal{P} \to \mathcal{P}'$ satisfying

1. $p = (p_1, \ldots, p_r) \in \mathcal{R} \Leftrightarrow \phi(p) = (\phi(p_1), \ldots, \phi(p_r)) \in \mathcal{R}'$
2. $\alpha(p) = \alpha'(\phi(p))$ for all $p \in \mathcal{R}$.

The *isomorphism* class $[X]$ of X consists of all $\mathcal{A}$-attributed r-structures on $\mathcal{P} = \mathbb{Z}_n$ that are isomorphic to X. By $\mathcal{S}_{\mathcal{A}}^{n,r}$ we denote the set of all $\mathcal{A}$-attributed r-structures on $\mathcal{P} = \mathbb{Z}_n$ and by $\left[\mathcal{S}_{\mathcal{A}}^{n,r}\right]$ the set of all isomorphism classes of structures from $\mathcal{S}_{\mathcal{A}}^{n,r}$.

We can identify any r-structure $X = (\mathbb{Z}_m, \mathcal{R}, \alpha)$ of order $m < n$ with a structure of order n by adding $q = n - m$ isolated points. The aligned structure is then of the form $X' = (\mathbb{Z}_n, \mathcal{R}, \alpha')$, where

$$\alpha'(p) = \begin{cases} \alpha(p) & : \quad p \in \mathcal{R} \\ 0 & : \quad \textit{otherwise} \end{cases} .$$

Using alignment, we can regard $\mathcal{S}_{\mathcal{A}}^{n,r}$ as the set of $\mathcal{A}$-attributed r-structures of bounded order n. Similarly, we may think of $\left[\mathcal{S}_{\mathcal{A}}^{n,r}\right]$ as the set of abstract $\mathcal{A}$-attributed r-structures of bounded order n. Again recall that specifying a bound n and aligning smaller structures to structures of order n are purely technical assumptions to simplify mathematics.

3.2 $\mathcal{T}$-Spaces

Let $\mathcal{X} = \mathbb{R}^n$ be the n-dimensional Euclidean vector space, and let $\mathcal{T}$ be a subgroup of the group of all $n \times n$ permutation matrices. Then the binary operation

$$\cdot : \mathcal{T} \times \mathcal{X} \to \mathcal{X}, \quad (T, x) \mapsto Tx$$

is a group action of $\mathcal{T}$ on $\mathcal{X}$. For $x \in \mathcal{X}$, the *orbit* of x is denoted by

$$[x]_{\mathcal{T}} = \{Tx : T \in \mathcal{T}\}.$$

If no misunderstanding can occur, we write $[x]$ instead of $[x]_{\mathcal{T}}$.

A *$\mathcal{T}$-space* over $\mathcal{X}$ is the orbit space $\mathcal{X}_{\mathcal{T}} = \mathcal{X}/\mathcal{T}$ of all orbits of $x \in \mathcal{X}$ under the action of $\mathcal{T}$. We call $\mathcal{X}$ the *representation space* of $\mathcal{X}_{\mathcal{T}}$. By

$$\mu : \mathcal{X} \to \mathcal{X}_{\mathcal{T}}$$

we denote the *membership function* that sends vector representations to the structure they describe.

$\mathcal{T}$-spaces are a convenient abstraction of r-structures in order to adopt geometrical and analytical concepts. To see this, let $\mathcal{A} = \mathbb{R}^d$ and let $\mathcal{X} = \mathcal{A}^N$, where $N = n^r$. Via the matrix representations,

we can identify r-structures from $\mathcal{S}_{\mathcal{A}}^{n,r}$ as vectors from $\mathcal{X}$. Obviously, we have a relaxation in the sense that $\mathcal{S}_{\mathcal{A}}^{n,r} \subseteq \mathcal{X}$ and $\left[\mathcal{S}_{\mathcal{A}}^{n,r}\right] \subseteq \mathcal{X}_{\mathcal{T}}$ such that μ restricted on $\mathcal{S}_{\mathcal{A}}^{n,r}$ sends vector representations to the structures they represent. Note that there are structures in $\mathcal{X}_{\mathcal{T}}$ that are not well-defined r-structures from $\left[\mathcal{S}_{\mathcal{A}}^{n,r}\right]$. Hence, care must be taken when applying $\mathcal{T}$-spaces.

The following notations, definitions, and results are useful to simplify technicalities. We use capital letters $X, Y, Z, \ldots$ to denote the elements of $\mathcal{X}_{\mathcal{T}}$. Suppose that $X = \mu(x)$ for some $x \in \mathcal{X}$. Then we identify X with $[x]$ and make use of sloppy notations like, for example, $x \in X$ to denote $x \in [x]$.[4]

Let $f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function satisfying $f(x,y) = f(y,x)$ for all $x, y \in \mathcal{X}$. Then f induces symmetric functions

$$F^* : \mathcal{X}_{\mathcal{T}} \times \mathcal{X}_{\mathcal{T}} \to \mathbb{R}, \quad (X,Y) \mapsto \max\{f(x,y) : x \in X, y \in Y\},$$
$$F_* : \mathcal{X}_{\mathcal{T}} \times \mathcal{X}_{\mathcal{T}} \to \mathbb{R}, \quad (X,Y) \mapsto \min\{f(x,y) : x \in X, y \in Y\}.$$

Since $\mathcal{T}$ is finite, the orbits $[x]$ of x are finite. Hence, F^* and F_* assume an extremal value. We call F^* *maximizer* and F_* *minimizer* of f on $\mathcal{X}_{\mathcal{T}} \times \mathcal{X}_{\mathcal{T}}$.

An inner product $\langle \cdot, \cdot \rangle$ on $\mathcal{X}$ gives rise to a maximizer of the form

$$\langle \cdot, \cdot \rangle^* : \mathcal{X}_{\mathcal{T}} \times \mathcal{X}_{\mathcal{T}} \to \mathbb{R}, \quad (X,Y) \mapsto \max\{\langle x,y \rangle : x \in X, y \in Y\}.$$

We call $\langle \cdot, \cdot \rangle^*$ *inner $\mathcal{T}$-product* induced by $\langle \cdot, \cdot \rangle$. The inner $\mathcal{T}$-product is *not* an inner product, because the maximum-operator in the definition of $\langle \cdot, \cdot \rangle^*$ does not preserve the bilinearity property of an inner product. But we can show that an inner $\mathcal{T}$-product satisfies some weaker properties.

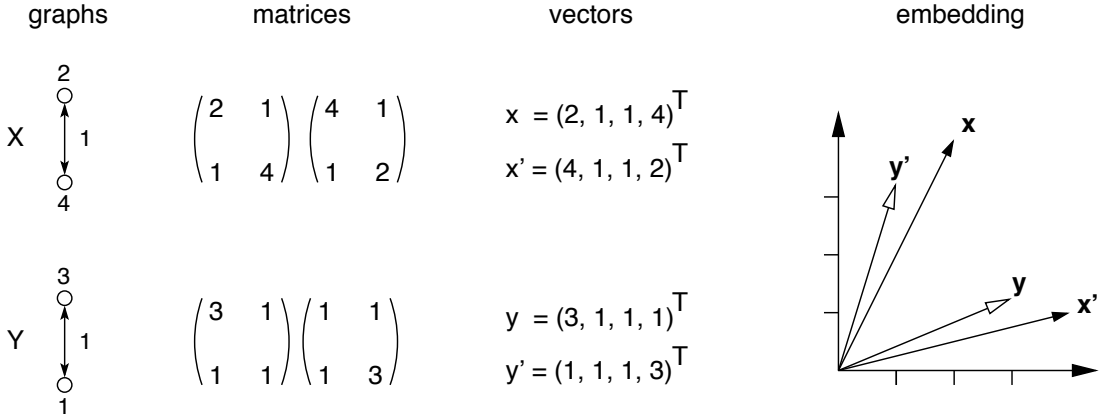


Figure 5: Illustration of two example graphs and their embeddings in a vector space (see Figure 4 for a detailed description).

Example 6 *Consider the graphs X and Y from Figure 5. We have*

$$\langle X,Y \rangle^* = \langle x,y' \rangle = \langle x',y \rangle = 16.$$

4. The notation is sloppy, because X is an element in $\mathcal{X}_{\mathcal{T}}$ and not a set, whereas $[x]$ is a set of equivalent elements from $\mathcal{X}$.

Thus, to determine $\langle X,Y\rangle^*$, we select vector representations $\tilde{x}$ of X and $\tilde{y}$ of Y that have closest angle and then evaluate $\langle \tilde{x},\tilde{y}\rangle$.

Any inner product space $\mathcal{X}$ is a normed space with norm $\|x\| = \sqrt{\langle x,x\rangle}$ and a metric space with metric $d(x,y) = \|x-y\|$. The norm $\|\cdot\|$ and the metric d on $\mathcal{X}$ give rise to minimizers $\|\cdot\|_*$ of $\|\cdot\|$ and D_* of d on $\mathcal{X}_\mathcal{T}$.

Since elements from $\mathcal{T}$ preserve lengths and angles, we have

$$\|Tx\| = \|Tx-0\| = \|Tx-T0\| = \|x-0\| = \|x\|$$

for all $T \in \mathcal{T}$. Hence, $\|X\|_*$ is independent from the choice of vector representation. We call the minimizer $\|\cdot\|_*$ the *$\mathcal{T}$-norm* induced by the norm $\|\cdot\|$. A $\mathcal{T}$-norm is related to an inner $\mathcal{T}$-product in the same way as a norm to an inner product. We have

1. $\langle X,X\rangle^* = \langle x,x\rangle$ for all $x \in X$.
2. $\|X\|_* = \sqrt{\langle X,X\rangle^*}$.

Note that a $\mathcal{T}$-norm is *not* a norm, because a $\mathcal{T}$-space has no well defined addition. But we can show that a $\mathcal{T}$-norm has norm-like properties.

Example 7 *Consider the graphs X and Y from Figure 5. To determine their $\mathcal{T}$-norm, it is sufficient to compute the standard norm of an arbitrarily chosen vector representation. Hence, we have*

$$\|X\|_* = \|x\| = \|x'\| = \sqrt{22},$$
$$\|Y\|_* = \|y\| = \|y'\| = \sqrt{12}.$$

The minimizer D_* of the Euclidean metric $d(x,y) = \|x-y\|$ is also a metric. To distinguish from ordinary metrics, we call the minimizer D_* of a Euclidean metric d on $\mathcal{X}$ the $\mathcal{T}$-metric induced by d. We can express the metric D_* in terms of $\langle\cdot,\cdot\rangle^*$ as follows:

$$D_*(X,Y)^2 = \|X\|_*^2 - 2\langle X,Y\rangle^* + \|Y\|_*^2.$$

Example 8 *Consider the graphs X and Y depicted in Figure 5. To determine $D_*(X,Y)$, we select vector representations $\tilde{x}$ of X and $\tilde{y}$ of Y that have minimal distance $d(\tilde{x},\tilde{y})$. Then we find that*

$$D_*(X,Y) = d(x,y') = d(x',y) = \sqrt{2}.$$

3.3 Functions on $\mathcal{T}$-Spaces

A *$\mathcal{T}$-function* is a function of the form

$$F : \mathcal{X}_\mathcal{T} \to \mathbb{R},$$

where $\mathcal{X}_\mathcal{T}$ is a T-space over $\mathcal{X}$. Instead of considering the $\mathcal{T}$-function F, it is often more convenient to consider its *representation function*

$$f : \mathcal{X} \to \mathbb{R}, \quad x \mapsto F \circ \mu(x),$$

which is invariant under transformations from elements of $\mathcal{T}$.

Here, the focus is on $\mathcal{T}$-functions that are locally Lipschitz. A $\mathcal{T}$-function is *locally Lipschitz* if, and only if, its representation function is locally Lipschitz. We refer to Appendix C for basic definitions and properties from nonsmooth analysis of locally Lipschitz functions.

Suppose that F is a locally Lipschitz function with representation function f. By Rademacher's Theorem 23, f is differentiable almost everywhere. In addition, at non-differentiable points, f admits the concept of generalized gradient. The concepts differentiability and gradient can be transfered to $\mathcal{T}$-spaces in a well-defined way. Assume that f is differentiable at some point $x \in \mathcal{X}$ with gradient $\nabla f(x)$. Then f is differentiable at all points $Tx \in X$ with $T \in \mathcal{T}$ and the gradient of f at Tx is of the form

$$\nabla f(Tx) = T\nabla f(x).$$

We say F is $\mathcal{T}$-differentiable at X, if its representation function f is differentiable at an arbitrary vector representation $x \in X$. The well-defined structure

$$\nabla F(X) = \mu(f(x))$$

is the *$\mathcal{T}$-gradient* of F at X pointing in direction of steepest ascent.

3.4 Optimization of Locally Lipschitz $\mathcal{T}$-Functions

A standard technique in machine learning and pattern recognition is to pose a learning problem as an optimization problem. Here, we consider the problem of solving optimization problems of the form

$$(P1) \qquad \begin{array}{ll} \text{minimize} & F : \mathcal{X}_{\mathcal{T}} \to \mathbb{R}, \qquad X \mapsto \sum_{i=1}^{k} F_i(X) \\ \text{subject to} & X \in \mathcal{U}_{\mathcal{T}} \end{array}$$

where the component functions F_i are locally Lipschitz $\mathcal{T}$-functions and $\mathcal{U}_{\mathcal{T}} \subset \mathcal{X}_{\mathcal{T}}$ is the feasible set of admissible solutions. Then according to Prop. 21, the cost function F is also locally Lipschitz, and we can rewrite (P1) to an equivalent optimization problem

$$(P2) \qquad \begin{array}{ll} \text{minimize} & f : \mathcal{X} \to \mathbb{R}, \qquad x \mapsto \sum_{i=1}^{k} f_i(x) \\ \text{subject to} & x \in \mathcal{U} \end{array}$$

where the component functions f_i are the representation functions of F_i and $\mathcal{U} \subseteq \mathcal{X}$ is the feasible set with $\mu(\mathcal{U}) = \mathcal{U}_{\mathcal{T}}$. Hence, f is the representation function of the locally Lipschitz $\mathcal{T}$-function F and therefore also locally Lipschitz.

To minimize locally Lipschitz functions, the field of nonsmooth optimization offers a number of techniques. A survey of classical methods can be found in Mäkelä and Neittaanmäki (1992); Shor (1985). As an example, we describe subgradient methods, which are easy to implement and well-suited to identify the difficulties arising in nonsmooth optimization. Algorithm 1 outlines the basic procedure:

Algorithm 1 (Basic Incremental Algorithm)

choose starting point $x_1 \in \mathcal{U}$ and set $t := 0$
repeat
  set $\tilde{x}_{t,1} := x_1$
  for $i = 1, \ldots, k$ **do**
    `direction finding`:
      determine $d_{t,i} \in \mathcal{X}$ and $\eta > 0$ s.t. $\tilde{x}_{t,i} + \eta d_{t,i} \in \mathcal{U}$ and

$$f_i(\tilde{x}_{t,i} + \eta d_{t,i}) < f_i(\tilde{x}_{t,i})$$

    `line search`:
      find step size $\eta_{t,i} > 0$ such that $\tilde{x}_{t,i} + \eta_{t,i} d_{t,i} \in \mathcal{U}$ and

$$\eta_{t,i} \approx \arg\min_{\eta > 0} f_i(\tilde{x}_{t,i} + \eta d_{t,i})$$

    `updating`:
      set $\tilde{x}_{t,i+1} := \tilde{x}_{t,i} + \eta_{t,i} d_{t,i}$

  Set $x_{t+1} := \tilde{x}_{t,k+1}$
  Set $t := t + 1$
until some termination criterion is satisfied

To explain the algorithm, we first consider the case that f is smooth and $\mathcal{U} = \mathcal{X}$. In the step *direction finding*, we generate a descent direction by exploiting the fact that the direction opposite to the gradient is locally the steepest descent direction. Line search usually employs some efficient univariate smooth optimization method or polynomial interpolation. The necessary condition for a local minimum yields a termination criterion. Now suppose that f is locally Lipschitz. Then f admits a generalized gradient at each point. The generalized gradient coincides with the gradient at differentiable points and is a convex set of subgradients at non-differentiable points. For more details, we refer to Appendix C.

Subgradients, the elements of a generalized gradient, play a very important role in algorithms for non-differentiable optimization. The basic idea of subgradient methods is to generalize the methods for smooth problems by replacing the gradient by an arbitrary subgradient. In the direction finding step, Algorithm 1 computes an arbitrary subgradient $d \in \partial f(x)$ at the current point x. If f is differentiable at x, then the subgradient d coincides with the gradient $\nabla f(x)$. If in addition $d \neq 0$, then the opposite direction $-d$ is the direction of steepest descent. On the other hand, if f is not differentiable at x, then $-d$ is not necessarily a direction of descent of f at x. But since f is differentiable almost everywhere by Rademacher's Theorem 23, the set of non-differentiable points is a set of Lebesgue measure zero.

Line search uses predetermined step sizes $\eta_{t,i}$, instead of an exact or approximate line search as in the gradient method. One reason for this is that the direction $-d$ computed in the direction finding step is not necessarily a direction of descent. Thus, the viability of subgradient methods depend critically on the sequence of step sizes. One common choice are step sizes that satisfy

$$\sum_{t=0}^{\infty} \eta_t = \infty \quad \text{and} \quad \sum_{t=0}^{\infty} \eta_t^2 < \infty,$$

where $\eta_t = \eta_{t,1}$.

To formulate a termination criterion, we could—in principle—make use of the following necessary condition of optimality.

Theorem 1 *Let $f : \mathcal{X} \to \mathbb{R}$ be locally Lipschitz at its minimum (maximum) $x \in \mathcal{R}$. Then*

$$0 \in \partial f(x).$$

At non-differentiable points, however, an arbitrary subgradient provides no information about the existence of the zero in the generalized gradient $\partial f(x)$. Therefore, when assuming an instantly decreasing step size, one reliable termination criterion stops the algorithm as soon as the step size falls below a predefined threshold.

Since the subgradient method is not a descent method, it is common to keep track of the best point found so far, which is the one with smallest function value. For further advanced and more sophisticated techniques to minimize locally Lipschitz functions, we refer to Mäkelä and Neittaanmäki (1992); Shor (1985).

We conclude this section with a remark on determining intractable subgradients in a practical setting.

3.4.1 APPROXIMATING SUBGRADIENTS

Nonsmooth optimization as discussed in Mäkelä and Neittaanmäki (1992); Shor (1985) assumes that at each point x we can evaluate at least one subgradient $y \in \partial f(x)$ and the function value $f(x)$. In principle, this should be no obstacle for the class of problems we are interested in. In a practical setting, however, evaluating a subgradient as descent direction can be computationally intractable. For example, the pattern recognition problems described later in Section 4.2-4.5 are all computationally efficient for structures like point patterns, but NP-hard for structures like graphs. A solution to this problem is to approximate a subgradient by using polynomial time algorithms. An approximated subgradient corresponds to a direction that is no longer a subgradient of the cost function. In particular, at smooth points, an approximated (sub)gradient (hopefully) corresponds to a descent direction close to the direction of steepest descent. We call Algorithm 1 an *approximate incremental subgradient methods* if the direction finding step produces directions that are not necessarily subgradients of the corresponding component function f_i.

We replace the subgradient by a computationally cheaper approximation as a direction of descent. In a computer simulation, we show that determining a sample mean of weighted graph is indeed possible when using approximate subgradient methods.

Suppose that $\mathcal{X}_\mathcal{T}$ is the $\mathcal{T}$-space of simple weighted graphs over $\mathcal{X} = \mathbb{R}^{n\times n}$, and let $\mathcal{U}_\mathcal{T} \subseteq \mathcal{X}_\mathcal{T}$ be the subset of weighted graphs with attributes from the interval $[0,1]$. Our goal is to determine a sample mean of a collection of simple weighted graphs $\mathcal{D} = \{\mathcal{X}_1, \ldots, \mathcal{X}_k\} \subseteq \mathcal{U}_\mathcal{T}$.

Given a representation x of X, the computationally intractable task is to find a representation x_i of X_i such that $(x, x_i) \in \operatorname{supp}\left(d^2_{X_i}|x\right)$. This problem is closely related to the problem of computing the distance $D_*(X, X_i)$, which is known to be an NP-complete *graph matching problem*. Hence, in a practical setting, exact algorithms that guarantee to find a subgradient as descent direction are useless for all but the smallest graphs. A solution to this problem is to approximate a subgradient by using polynomial time algorithms. An approximated subgradient corresponds to a direction that is no longer a subgradient of the cost function. In particular, at smooth points, an approximated (sub)gradient (hopefully) corresponds to a descent direction close to the direction of steepest

descent. We call Algorithm 1 an *approximate incremental subgradient methods* if the direction finding step produces directions that are not necessarily subgradients of the corresponding component function f_i.

4. Pattern Recognition in $\mathcal{T}$-Spaces

This section shows how the framework of $\mathcal{T}$-spaces can be applied to solve problems in structural pattern recognition. We first propose a generic scheme for learning in distance spaces. Based on this generic scheme, we derive cost functions for determining a sample mean, central clustering, learning large margin classifiers, supervised learning in structured input and/or output spaces, and finding frequent substructures. Apart from the last problem, all other cost functions presented in this section extend standard cost functions from the vector space formalism to $\mathcal{T}$-spaces in the sense that we recover the standard formulations when regarding vectors as r-structures.

4.1 A Generic Approach: Learning in Distance Spaces

Without loss of generality, we may assume that $(\mathcal{X}_\mathcal{T}, D)$ is a distance space, where D is either the metric D_* induced by the Euclidean metric on $\mathcal{X}$ or another (not necessarily metric) distance function that is more appropriate for the problem to hand. A generic approach to solve a learning problem (P) in $\mathcal{X}_\mathcal{T}$ is as follows:

1. Transform (P) to an optimization problem, where the cost function F is a function defined on $\mathcal{X}_\mathcal{T}$.
2. Show that F is locally Lipschitz.
3. Optimize F using methods from nonsmooth optimization.

Since $\mathcal{X}_\mathcal{T}$ is a metric space over an Euclidean vector space, we can apply subgradient methods or other techniques from nonsmooth optimization to minimize locally Lipschitz $\mathcal{T}$-functions on $\mathcal{X}_\mathcal{T}$. If the cost function F depends on a distance measure D, we demand that D is locally Lipschitz to ensure the locally Lipschitz property for F.

4.2 The Sample Mean of k-Structures

The sample mean of structures is a basic concept for a number of methods in statistical pattern recognition. Examples include visualizing or comparing two populations of chemical graphs, and central clustering of structures (Section 4.3).

We define a sample mean of the k elements $X_1, \ldots, X_k \in \mathcal{X}_\mathcal{T}$ as a minimizer of

$$\begin{array}{ll} \text{minimize} & F(X) = \sum_{i=1}^{k} D(X, X_i)^2 \\ \text{subject to} & X \in \mathcal{X}_\mathcal{T} \end{array},$$

where D is a distance function on $\mathcal{X}_\mathcal{T}$. If D is locally Lipschitz, then F is also locally Lipschitz by Prop. 21.

First approaches to study averages of graphs have been pursued by Jiang, Münger, and Bunke (2001). They considered the set median and generalized median of a sample of graphs as a discrete optimization problem with a similar cost function as for the sample mean. To minimize the cost

function, they applied a genetic algorithm to graphs with a small number of discrete attributes. In this contribution, we shift the problem of determining a sample mean from discrete to continuous optimization.

4.3 Central Clustering of k-Structures

Suppose that we are given a training sample $\mathcal{X} = \{X_1, \ldots, X_m\}$ consisting of m structures X_i drawn from the structure space $\mathcal{X}_\mathcal{T}$. The aim of central clustering is to find k cluster centers $\mathcal{Y} = \{Y_1, \ldots, Y_k\} \subseteq \mathcal{X}_\mathcal{T}$ such that the following cost function

$$F(M, \mathcal{Y}, \mathcal{X}) = \frac{1}{m} \sum_{j=1}^{k} \sum_{i=1}^{m} m_{ij} D(X_i, Y_j),$$

is minimized with respect to a given distortion measure D. The matrix $M = (m_{ij})$ is a $(m \times k)$-membership matrix with elements $m_{ij} \in [0,1]$ such that $\sum_j m_{ij} = 1$ for all $i = 1, \ldots, m$.

If the distortion measure is locally Lipschitz, then F as a function of the cluster centers Y_j is locally Lipschitz by Prop. 21.

A number of central clustering algorithms for graphs have been devised recently (Gold, Rangarajan, and Mjolsness, 1996; Günter and Bunke, 2002; Lozano and Escolano, 2003; Jain and Wysotzki, 2004; Bonev, Escolano, Lozano, Suau, Cazorla, and Aguilar, 2007). In experiments it has been shown that the proposed methods converge to satisfactory solutions, although neither the notion of cluster center nor the update rule of the cluster centers is well-defined. Because of these issues one might expect that central clustering algorithm could be prone to oscillations halfway between different cluster centers of the same cluster. An explanation why this rarely occurs can now be given. As long as the cost function is locally Lipschitz, almost all points are differentiable. For these points the update rule is well-defined. Hence, it is very unlikely that the aforementioned oscillations occur over a longer period of time, when using an optimization algorithm that successively decreases the step size.

4.4 Large Margin Classifiers

Consider the function

$$h_{W,b} : \mathcal{X}_\mathcal{T} \to \mathbb{R}, \quad X \mapsto \langle W, X \rangle^* + b,$$

where $W \in \mathcal{X}_\mathcal{T}$ is the *weight structure* and $b \in \mathbb{R}$ the *bias*. The discriminant $h_{W,b}$ implements a two-category classifier in the obvious way: Assign an input structure X to the class labeled $+1$ if $h_{W,b}(X) \geq 0$ and to -1 if $h_{W,b}(X) < 0$.

Suppose that $\mathcal{Z} = \{(X_1, y_1), \ldots, (X_k, y_k)\}$ is a training sample consisting of k training structures $X_i \in \mathcal{X}_\mathcal{T}$ together with corresponding labels $y_i \in \{\pm 1\}$. We say, $\mathcal{Z}$ is *$\mathcal{T}$-separable* if there exists a $W_0 \in \mathcal{X}_\mathcal{T}$ and $b_0 \in \mathbb{R}$ with

$$h_*(X) = \langle W_0, X \rangle^* + b_0 = y$$

for all $(X, y) \in \mathcal{Z}$.

To find an "optimal" discriminant that correctly classifies the training examples, we construct the cost function

$$F(W, b, \alpha) = \frac{1}{2} \|W\|_*^2 - \sum_{i=1}^{k} \alpha_i \left(y_i \left(\langle W, X_i \rangle^* + b \right) - 1 \right),$$

where the $\alpha_i \geq 0$ are the Lagrangian multipliers. The representation function of F is of the form

$$f(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{k} \alpha_i y_i (s_i(w,b) - 1),$$

where $s_i(w,b) = \max_{T \in \mathcal{T}} \langle w, Tx_i \rangle + b$. The elements $w \in W$ and $x_i \in X_i$ are arbitrary. The first term of f is smooth and convex (and therefore locally Lipschitz). The locally Lipschitz property and convexity of the second term follows from the rules of calculus for locally Lipschitz functions (see Section C) and Prop. 20. Hence, f is locally Lipschitz and convex.

The structurally linear discriminants sets the stage to (i) explore large margin classifiers in structure spaces and (ii) construct neural learning machines for adaptive processing of finite structures. Subgradient methods for maximum margin learning has been applied in Ratliff, Bagnell, and Zinkevich (2006) for predicting structures rather than classes. Finally note that the inner $\mathcal{T}$-product as a maximizer of a set of similarities is not a kernel (Gärtner, 2005).

4.5 Supervised Learning

The next application example generalizes the problem of learning large margin classifiers for k-structures by allowing $\mathcal{T}$-spaces as input and as output space. Note that the in- and output spaces may consist of different classes of k-structures, for example, the input patterns can be feature vectors and the output space can be the domain of graphs.

Assume that we are given a a training sample $\mathcal{Z} = \{(X_1, Y_1), \ldots, (X_k, Y_k)\}$ consisting of k training structures X_i drawn from some $\mathcal{T}$-space $\mathcal{X}_\mathcal{T}$ over $\mathcal{X}$ together with corresponding output structures Y_i from a $\mathcal{T}'$-space $\mathcal{Y}_{\mathcal{T}'}$ over $\mathcal{Y}$. Given the training data $\mathcal{Z}$, our goal is to find an unknown functional relationship (hypothesis)

$$H : \mathcal{X}_\mathcal{T} \to \mathcal{Y}_\mathcal{T}$$

from a hypothesis space $\mathcal{H}$ that best predicts the output structures of unseen examples $(X,Y) \in \mathcal{X}_\mathcal{T} \times \mathcal{Y}_\mathcal{T}$ according to some cost function

$$F(H,\mathcal{Z}) = \frac{1}{k}\sum_{i=1}^{k} L(H(X_i), Y_i),$$

where $L : \mathcal{Y}_\mathcal{T} \times \mathcal{Y}_\mathcal{T} \to \mathbb{R}$ denotes the loss function.

The representation function of F is of the form

$$f(h,\mathcal{Z}) = \frac{1}{k}\sum_{i=1}^{k} \ell(h(x_i), y_i),$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is the representation function of L, $h : \mathcal{X} \to \mathcal{Y}$ the representation function of H, and $x_i \in X_i$, $y_i \in Y_i$. We assume that the functions h have a parametric form and are therefore uniquely determined by the value of their parameter vector θ_h. We make this dependence of h on θ_h explicit by writing $f(\theta_h, \mathcal{Z})$ instead of $f(h,\mathcal{Z})$.

The function f is locally Lipschitz if ℓ and h (as a function of θ_h) are locally Lipschitz. As an example for a locally Lipschitz function f, we extend supervised neural learning machines to k-structures:

- *Loss function*: The loss
$$\ell(x,y) = D_*(\mu(x),\mu(y))^2$$
is locally Lipschitz as a function of x.

- *Hypothesis space*: Consider the set $\mathcal{H}_{NN}$ of all functions $g : \mathcal{X} \to \mathcal{Y}$ that can be implemented by a neural network. Suppose that all functions from $\mathcal{H}_{NN}$ are smooth. If $\dim(\mathcal{V}_\mathcal{Y}) = M$, then g is of the form $g = (g_1,\ldots,g_M)$, where the g_i are the component functions of g. For each component g_i, the pointwise maximizer
$$h_i(x) = \max_{T\in\mathcal{T}} g_i(Tx)$$
is locally Lipschitz. Hence, $h = (h_1,\ldots,h_M)$ is locally Lipschitz.

Compared to common models in predicting structures as applied by Taskar (2004); Tsochantaridis, Hofmann, Joachims, and Altun (2004), the proposed approach differs in two ways: First, the proposed cost function requires no indirection via a score function $f : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ to select the prediction from $\mathcal{Y}$ by maximizing f for a given input from $\mathcal{X}$. Second, the proposed approach suggests a formulation that can be exploited to approximately solve discrete and continuous prediction problems.

4.6 Frequent Substructures

Our aim is to find the most frequent substructure occurring in a finite data set $\mathcal{D}$ of k-structures. To show how to apply the theory of $\mathcal{T}$-spaces to this problem, we consider a simplified setting.

First we define what we mean by a substructure. A k-structure $X' = (\mathbb{Z}_m,\mathcal{R}',\alpha')$ is said to be a *substructure* of a k-structure $X = (\mathbb{Z}_n,\mathcal{R},\alpha)$, if there is an isomorphic embedding $\phi : \mathbb{Z}_m \to \mathbb{Z}_n$.

Next, we restrict ourselves to k-structures with attributes from $[0,1] \subseteq \mathbb{R}$ for the sake of simplicity. Let
$$\mathcal{B}_\mathcal{T} = \big\{X = (\mathbb{Z}_n,\mathcal{R}_i,\alpha) \in \mathcal{X}_\mathcal{T} : \alpha(X) \subseteq \{0,1\}\big\},$$
$$\mathcal{U}_\mathcal{T} = \big\{X = (\mathbb{Z}_n,\mathcal{R}_i,\alpha) \in \mathcal{X}_\mathcal{T} : \alpha(X) \subseteq [0,1]\big\}$$
be the set of all $\mathcal{A}$-attributed k-structures $X \in \mathcal{X}_\mathcal{T}$ with attributes from $\{0,1\}$ and $[0,1]$, respectively. Suppose that $\mathcal{D} = \{X_1,\ldots,X_k\} \subseteq \mathcal{B}_\mathcal{T}$ is a set of k-structures.

The characteristic function of the i-th structure $X_i \in \mathcal{D}$
$$\chi_i(X) = \begin{cases} 1 & : \quad X \text{ is a substructure of } X_i \\ 0 & : \quad \text{otherwise} \end{cases}.$$
indicates whether the k-structure X is a substructure of X_i. We say X^* is a *maximal frequent substructure* of order m if it solves the following discrete problem
$$\begin{array}{ll} \text{maximize} & F(X) = \sum_{i=1}^k \chi_i(X) \\ \text{subject to} & |X| = m \\ & X \in \mathcal{B}_\mathcal{T}. \end{array}$$
We cast the discrete to a continuous problem. For this, we define
$$F_i(X) = \frac{\langle X, X_i\rangle^*}{\|X\|_*^2}.$$

for all $X \in \mathcal{U}_\mathcal{T}$. We have $F_i(X) \in [0,1]$ with $F_i(X) = 1$ if, and only if, X is a substructure of X_i. Consider for a moment the problem to maximize the criterion function

$$G(X) = \sum_{i=1}^{k} F_i(X).$$

The problem with this criterion function is that a maximizer $X \in \mathcal{U}_\mathcal{T}$ of G could be a k-structure not occurring as a substructure in any of the k-structures from $\mathcal{D}$. To fix this problem, we use the soft-max function $\exp(\beta(F_i(X)-1))$ with control parameter β. In the limit $\beta \to \infty$, the i-th soft-max function reduces to the characteristic function χ_i. Given a fixed $\beta > 0$, the soft-max formulation of the frequent subgraph problem is of the form

$$\begin{array}{ll} \text{maximize} & F_\beta(X) = \sum_{i=1}^{k} \exp\{\beta(F_i(X)-1)\} \\ \text{subject to} & |X| = m \\ & X \in \mathcal{U}_\mathcal{T}. \end{array}$$

The representation function of F_β is of the form

$$f_\beta(x) = \sum_{i=1}^{k} \exp\left\{\beta\left(\frac{s_i(x)}{\|x\|} - 1\right)\right\},$$

where $s_i(x) = \max_{T \in \mathcal{T}} \langle x, Tx_i \rangle$. Applying the rules of calculus yields that f_β is locally Lipschitz.

The common approach casts the frequent subgraph mining problem to a search problem in a state space, which is then solved by a search algorithm (Dehaspe, Toivonen, and King, 1998; Han, Pei, and Yin, 2000; Inokuchi, Washio, and Motoda, 2000; Kuramochi and Karypis, 2001; Yan and Han, 2002). Here, we suggest a continuous cost function for the frequent subgraph mining problem that can be solved using optimization based methods (see Section 3.4).

5. Experimental Results

To demonstrate the effectiveness and versatility of the proposed framework, we applied it to the problem of determining a sample mean of randomly generated point patterns and weighted graphs as well as to central clustering of letters and protein structures represented by graphs.

5.1 Sample Mean

To assess the performance and to investigate the behavior of the subgradient and approximated subgradient method for determining a sample mean, we conducted an experiments on random graphs, letter graphs, and chemical graphs. For computing approximate subgradients we applied the graduated assignment (`GA`) algorithm (see Appendix D). For data sets consisting of small graphs, we also applied a depth first search (`DF`) algorithm that guarantees to return an exact subgradient.

5.1.1 DATA

Random Graphs. The first data set consists of randomly generated graphs. We sampled k graphs by distorting a given initial graph according to the following scheme: First, we randomly generated an initial graph M_0 with 6 vertices and edge density 0.5. Next, we assigned a feature vector to

	depth-first search	graduated assignment	set mean
Random Graphs	29.6 (± 5.3)	34.5 (± 6.6)	43.0 (± 7.5)
Letter Graphs	42.3 (± 10.1)	43.9 (± 11.1)	60.5 (± 16.6)
Molecules		262.2 (± 113.6)	338.0 (± 115.0)

Table 1: Average SSD of sample mean approximated by depth-first search and graduated assignment. As reference value the average SSD of the set mean is shown in the last column. Standard deviations are given in parentheses.

each vertex and edge of M_0 drawn from a uniform distribution over $[0,1]^d$ ($d = 3$). Given M_0, we randomly generated k distorted graphs as follows: Each vertex and edge was deleted with 20% probability. A new vertex was inserted with 10% probability and randomly connected to other vertices with 50% probability. Uniform noise from $[0,1]^d$ with standard deviation $\sigma \in [0,1]$ was imposed to all feature vectors. Finally, the vertices of the distorted graphs were randomly permuted.

We generated 500 samples each consisting of $k = 10$ graphs. For each sample the noise level $\sigma \in [0,1]$ was randomly prespecified.

Letter Graphs. The letter graphs were taken from the IAM Graph Database Repository. [5] The graphs represent distorted letter drawings from the Roman alphabet that consist of straight lines only. Lines of a letter are represented by edges and ending points of lines by vertices. Each vertex is labeled with a two-dimensional vector giving the position of its end point relative to a reference coordinate system. Edges are labeled with weight 1. We considered the 150 letter graphs representing the capital letter *A* at a medium distortion level.

We generated 100 samples each consisting or $k = 10$ letter graphs drawn from a uniform distribution over the data set of 150 graph letters representing letter *A* at a medium distortion level.

Chemical Graphs. The chemical compound database was taken from the gSpan site[6]. The data set contains 340 chemical compounds, 66 atom types, and 4 types of bonds. On average a chemical compound consists of 27 vertices and 28 edges. Atoms are represented by vertices and bonds between atoms by edges. As attributes for atom types and type of bonds, we used a 1-to-k binary encoding, where $k = 66$ for encoding atom types and $k = 4$ for encoding types of bonds.

We generated 100 samples each consisting of $k = 10$ chemical graphs drawn from a uniform distribution over the data set of 340 chemical graphs.

5.1.2 EVALUATION PROCEDURE

As performance measure, we used the average sum-of-squared distances (SSD) of the sample mean described in Section 4.2 averaged over all samples. The average SSD of the set mean graph serves as our reference value. The set mean is an element from the set $\mathcal{D}$ itself that minimizes the SSD over all structures from $\mathcal{D}$.

Algorithm 2 (K-Means for Structures)

initialize number k of clusters
initialize cluster centers $Y_1, \ldots, Y_k$
repeat
    classify structures X_i according to nearest Y_j
    recompute Y_j
until some termination criterion is satisfied

5.1.3 RESULTS

Table 1 shows the average SSD and its standard deviation. The results show that using exact subgradients gives better approximations of the sample mean than using approximated subgradients. Compared with the set median, the results indicate that the subgradient and approximated subgradient method have found reasonable solutions in the sense that the resulting average SSD is lower.

5.2 Central Clustering

Based on the concept of sample mean for structures, we applied the structural versions of k-means and simple competitive learning on four data sets in order to assess and compare the performance of subgradient methods.

5.2.1 CENTRAL CLUSTERING ALGORITHMS FOR GRAPHS

We consider k-means and simple competitive learning in order to minimize the cluster objective (see Section 4.2)[7]

$$F(M, \mathcal{Y}, \mathcal{X}) = \frac{1}{2} \sum_{j=1}^{k} \sum_{i=1}^{m} m_{ij} D(X_i, Y_j).$$

K-means for graphs. The structural version of k-means is outlined in Algorithm 2. This method operates as the EM algorithm of standard k-means, where the chosen distortion measure in the E-step is D to classify the structures X_i according to nearest cluster center Y_j . In the M-step the basic incremental subgradient method described in Algorithm 1 is applied to recompute the means.
Simple competitive learning. The structural version of simple competitive learning corresponds to the basic incremental subgradient method described in Algorithm 1 for minimizing the cluster objective $F(X)$.

5.2.2 DATA

We selected four data sets described in Riesen and Bunke (2008). The data sets are publicly available at the IAM Graph Database Repository. Each data set is divided into a training, validation, and a test set. In all four cases, we considered data from the test set only. The description of the data

5. The repository can be found at `http://www.iam.unibe.ch/fki/databases/iam-graph-database`.
6. gSpan can be found at `http://www.xifengyan.net/software/gSpan.htm`.
7. We replaced the factor $1/m$ by the factor $1/2$ for convenience of presentation of our results.

data set	#graphs)	#(classes)	avg(nodes)	max(nodes)	avg(edges)	max(edges)
letter	750	15	4.7	8	3.1	6
grec	528	22	11.5	24	11.9	29
fingerprint	900	3	8.3	26	14.1	48
molecules	100	2	24.6	40	25.2	44

Table 2: Summary of main characteristics of the data sets used for central clustering.

sets are mainly excerpts from Riesen and Bunke (2008). Table 2 provides a summary of the main characteristics of the data sets.

Letter Graphs. We consider all 750 graphs from the test data set representing distorted letter drawings from the Roman alphabet that consist of straight lines only (A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z). The graphs are uniformly distributed over the 15 classes (letters). The letter drawings are obtained by distorting prototype letters at low distortion level. Lines of a letter are represented by edges and ending points of lines by vertices. Each vertex is labeled with a two-dimensional vector giving the position of its end point relative to a reference coordinate system. Edges are labeled with weight 1. Figure 6 shows a prototype letter and distorted version at various distortion levels.

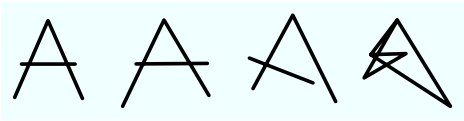


Figure 6: Example of letter drawings: Prototype of letter A and distorted copies generated by imposing low, medium, and high distortion (from left to right) on prototype A.

GREC Graphs. The GREC data set (Dosch and Valveny, 2006) consists of graphs representing symbols from architectural and electronic drawings. We use all 528 graphs from the test data set uniformly distributed over 22 classes. The images occur at five different distortion levels. In Figure 7 for each distortion level one example of a drawing is given. Depending on the distortion level, either erosion, dilation, or other morphological operations are applied. The result is thinned to obtain lines of one pixel width. Finally, graphs are extracted from the resulting denoised images by tracing the lines from end to end and detecting intersections as well as corners. Ending points, corners, intersections and circles are represented by vertices and labeled with a two-dimensional attribute giving their position. The vertices are connected by undirected edges which are labeled as line or arc. An additional attribute specifies the angle with respect to the horizontal direction or the diameter in case of arcs.

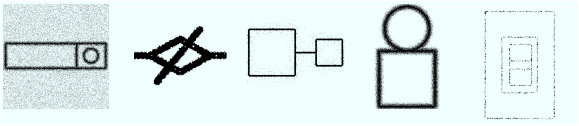


Figure 7: GREC symbols: A sample image of each distortion level

Fingerprint Graphs. We consider a subset of 900 graphs from the test data set representing fingerprint images of the NIST-4 database (Watson and Wilson, 1992). The graphs are uniformly distributed over three classes *left*, *right*, and *whorl*. A fourth class (*arch*) is excluded in order to keep the data set balanced. Fingerprint images are converted into graphs by filtering the images and extracting regions that are relevant (Neuhaus and Bunke, 2005). Relevant regions are binarized and a noise removal and thinning procedure is applied. This results in a skeletonized representation of the extracted regions. Ending points and bifurcation points of the skeletonized regions are represented by vertices. Additional vertices are inserted in regular intervals between ending points and bifurcation points. Finally, undirected edges are inserted to link vertices that are directly connected through a ridge in the skeleton. Each vertex is labeled with a two-dimensional attribute giving its position. Edges are attributed with an angle denoting the orientation of the edge with respect to the horizontal direction. Figure 8 shows fingerprints of each class.

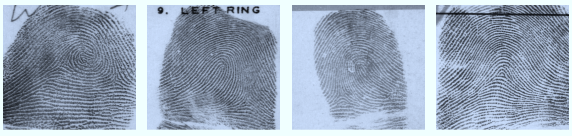


Figure 8: Fingerprints: (a) Left (b) Right (c) Arch (d) Whorl. Fingerprints of class arch are not considered.

Molecules. The mutagenicity data set consists of chemical molecules from two classes (mutagen, non-mutagen). The data set was originally compiled by Kazius, McGuire, and Bursi (2005) and reprocessed by Riesen and Bunke (2008). We consider a subset of 100 molecules from the test data set uniformly distributed over both classes. We describe molecules by graphs in the usual way: atoms are represented by vertices labeled with the atom type of the corresponding atom and bonds between atoms are represented by edges labeled with the valence of the corresponding bonds. We used a 1-to-k binary encoding for representing atom types and valence of bonds, respectively.

5.2.3 GENERAL EXPERIMENTAL SETUP

In all experiments, we applied k-means and simple competitive learning for graphs to the aforementioned data sets. We used the following experimental setup:
Performance measures. We used the following measures to assess the performance of an algorithm on a data set: (1) error value of the cluster objective (see Section 4.2), (2) classification accuracy, and (3) silhouette index. The silhouette index is a cluster validation index taking values from $[-1,1]$. Higher values indicate a more compact and well separated cluster structure. For more details we refer to Theodoridis and Koutroumbas (2009).
Initialization of the clustering algorithms. The number k of centroids as shown in Table 3 was chosen by compromising a satisfactory classification accuracy against the silhouette index. To initialize both clustering algorithms, we used a modified version of the "furthest first" heuristic (Hochbaum and Shmoys, 1985). For each data set $\mathcal{S}$, the first centroid Y_1 is initialized to be a graph closest to the sample mean of $\mathcal{S}$. Subsequent centroids are initialized according to

$$Y_{i+1} = \arg\max_{X \in \mathcal{S}} \min_{Y \in \mathcal{Y}_i} D(X,Y),$$

data set	k	measure	km	cl
letter	30			
		error	11.6	11.1
		accuracy	0.86	0.90
		silhouette	0.38	0.40
grec	33			
		error	32.7	27.6
		accuracy	0.84	0.87
		silhouette	0.40	0.42
fingerprint	60			
		error	1.88	1.30
		accuracy	0.81	0.79
		silhouette	0.32	0.34
molecules	10			
		error	56.0	53.8
		accuracy	0.68	0.70
		silhouette	0.04	0.05

Table 3: Results of k-means (km) and simple competitive learning (cl) on four data sets.

where $\mathcal{Y}_i$ is the set of the first i centroids chosen so far.

Subgradient and graph distance calculations. For subgradient and graph distance calculations, we applied a depth first search algorithm on the letter data set and the graduated assignment algorithm (Gold and Rangarajan, 1996) on the grec, fingerprint, and molecule data set.

5.2.4 RESULTS

Table 3 summarizes the results. The first observation to be made is that simple competitive learning performs slightly better than k-means with respect to all three performance measures. This is in contrast to findings on standard k-means and simple competitive learning in vector spaces. The second observation is that both k-means algorithms yield satisfying classification accuracies on all data sets. This result shows that approximated subgradient methods can be applied to central clustering in the domain of graphs.

5.3 Clustering Protein Structures

In our last experiment, we compared the performance of k-means and simple competitive learning of graphs with hierarchical clustering applied on protein structures.

5.3.1 DATA: PROTEIN CONTACT MAPS

One common way to model the 3D structure of proteins are contact maps. A contact map is a graph $X = (V,E)$ with ordered vertex set. Vertices represent residues. Two vertices are connected by an edge (contact) if the spatial distance of the corresponding residues is below some prespecified threshold.

ID	domain	ID	domain	ID	domain	ID	domain
1	1b00A	11	4tmyB	21	2b3iA	31	1tri
2	1dbwA	12	1rn1A	22	2pcy	32	3ypiA
3	1nat	13	1rn1B	23	2plt	33	8timA
4	1ntr	14	1rn1C	24	1amk	34	1ydvA
5	1qmpA	15	1bawA	25	1aw2A	35	1b71A
6	1qmpB	16	1byoA	26	1b9bA	36	1bcfA
7	1qmpC	17	1byoB	27	1btmA	37	1dpsA
8	1qmpD	18	1kdi	28	1htiA	38	1fha
9	3chy	19	1nin	29	1tmhA	39	1ier
10	4tmyA	20	1pla	30	1treA	40	1rcd

Table 4: PDB domain names of the Skolnick test set and their assigned indexes (ID).

No	Style	Residues	Seq. Sim.	Proteins
1	alpha-beta	124	15-30%	1-14
2	beta	99	35-90%	15-23
3	alpha-beta	250	30-90%	24-34
4		170	7-70%	35-40

Table 5: Characteristic properties of the Skolnick test set as taken from Caprara and Lancia (2002). Shown are the fold style, mean number of residues, and the range of similarity obtained by sequence alignment of the protein domains.

We used the Skolnick test set consisting of 40 protein contact maps provided by Xie and Sahinidis (2007). Table 4 shows the PDB domain names of the test set and their assigned indexes.[8] Table 5 describes characteristic properties of the protein domains. The characteristic feature of the Skolnick data is that sequence similarity fails for correct categorization of the proteins as indicated by the fourth column (*Seq. Sim.*) of Table 5. This motivates structural alignment for solving the Skolnick clustering test.

5.3.2 ALGORITHMS

To cluster the contact maps, we minimized the cluster objective described in Section 4.3

$$F(M,\mathcal{Y},\mathcal{X}) = \frac{1}{m}\sum_{j=1}^{k}\sum_{i=1}^{m} m_{ij} D(X_i, Y_j),$$

using the extensions of k-means and simple competitive learning. The chosen distance measure D for both, the letter graphs and the contact maps, is the minimizer of the standard Euclidean metric.

8. The Protein Data Bank (PDB) is a repository for the 3D structures of proteins, nucleic acids, and other large biological molecules.

C	ID	Fold	Superfamily	Family
1	1-11	Flavodin-like	Che Y-like	Che Y-related
2	12-14	Microbial ribonucl.	Microbial ribonucl.	Fungi ribonucl.
3	15-23	Cuperdoxin-like	Cuperdoxins	Plastocyanim-like Plastoazurin-like
4	24-34	TIM-beta alpha-barrel	Triosephosphate isomerase (TIM)	Triosephosphate isomerase (TIM)
5	35-40	Ferritin-like	Ferritin-like	Ferritin

Table 6: Clusters of Skolnick proteins detected by competitive learning and k-means. Shown are the cluster memberships of the proteins via their indexes (ID) as assigned in Table 5. The clusters perfectly agree with the fold, family, and superfamily according to SCOP categories.

For letter graphs the underlying transformation set $\mathcal{T}$ is the set of all possible vertex permutations. In the case of contact maps, the set $\mathcal{T}$ is the subset of all partial vertex permutations that preserve the order of the vertices.

For subgradient and distance calculations, we used a combination of graduated assignment and dynamic programming (Jain and Lappe, 2007).

5.3.3 RESULTS

Competitive learning and k-means both correctly categorized the 40 proteins in 5 clusters according to the SCOP categories as shown in Table 6.[9] This result was also achieved by previous approaches based on hierarchical clustering using pairwise similarity matrices (Xie and Sahinidis, 2007; Caprara and Lancia, 2002; Caprara, Carr, Istrail, Lancia, and Walenz, 2004).

Competitive learning and k-means require less pairwise structural alignments than pairwise clustering. Pairwise clustering of 40 structures requires 780 structural alignments. In contrast, competitive learning required 120 and k-means 440 structural alignments. One problem of central clustering algorithms applied to contact maps is the increasing size of the cluster centers caused by the updating step. A solution to this problem is to restrict the vertices of a cluster center to those vertices that occur in at least one cluster member. In doing so, spurious vertices of former cluster members are removed.

The distinguishing feature of central clustering of structures is that we obtain prototypes for each cluster. According to Jain and Obermayer (2009), the sample mean is equivalent to the multiple alignment of proteins, which is like clustering an essential task in bioinformatics. Hence, central clustering of protein structures can solve two tasks simultaneously, categorizing the proteins and and multiple aligning cluster members, which is useful for protein structure classification, structure-based function prediction, and highlighting structurally conserved regions of functional significance.

9. The *Structural Classification of Proteins* (SCOP) database is a largely manual classification of protein structural domains based on similarities of their amino acid sequences and 3D structures.

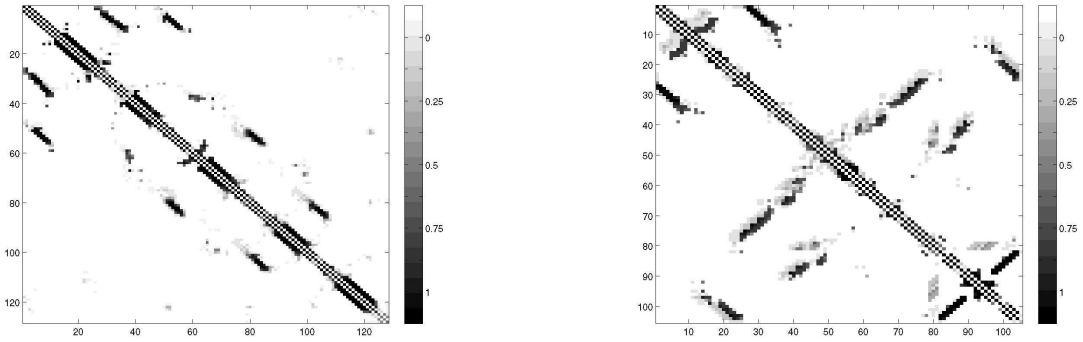


Figure 9: Shown are approximated sample means of the Che Y-like superfamily of cluster C1 (**left**) and the Cuperdoxins superfamily family of cluster C3 (**right**). Diagonal elements show the residues and off-diagonal elements the contacts. Darker shading refers to a higher relative frequency of occurrence of residues/contacts over all cluster members.

Figure 9 shows approximations of sample means of the two largest clusters computed by competitive learning.

6. Summary

In this contribution, we described a generic technique of how to generalize classical learning approaches and other problems from pattern recognition to structured domains. The proposed technique is based on the notion of $\mathcal{T}$-space. A $\mathcal{T}$-space is a quotient set of a metric vector space—the representation space—with all the vectors identified that represent the same structure. The equivalence classes of representation vectors are determined by the subgroup of homogeneous isometrics $\mathcal{T}$. This constructions turns out to be a convenient abstraction of combinatorial structures to formally adopt geometrical and analytical concepts from vector spaces.

The metric of the representation space $\mathcal{X}$ induces a metric on the $\mathcal{T}$-space. A norm on $\mathcal{X}$ induces a $\mathcal{T}$-norm on $\mathcal{X}_\mathcal{T}$. The $\mathcal{T}$-norm corresponds to the same geometric concept of length as the standard norm. Thus, different vector representations of the same structure have the same length. An inner product $\langle\cdot,\cdot\rangle$ on $\mathcal{X}$ induces the inner $\mathcal{T}$-product on $\mathcal{X}_\mathcal{T}$, which is not bilinear but has the same geometrical properties as $\langle\cdot,\cdot\rangle$. In other words, the Cauchy-Schwarz inequality is valid for structures. This result gives rise to a well-defined geometric concept of angle between structures.

The interplay of geometrical intuition, the algebraic group structure of the transformation set $\mathcal{T}$, and the link to the properties of a vector space via the membership function yields the well-defined notions of $\mathcal{T}$-differentiability and $\mathcal{T}$-gradient that generalize the standard definitions of differentiability and gradient of a smooth function at some point from a vector space. In particular, the $\mathcal{T}$-gradient of a $\mathcal{T}$-function at a $\mathcal{T}$-differentiable point is a well-defined structure pointing in direction of steepest ascent and satisfies the necessary condition of optimality. Therefore, we can apply local gradient information to minimize smooth $\mathcal{T}$-functions.

One application of the theory of $\mathcal{T}$-spaces are problems from structural pattern recognition. For selected problems, we presented continuous optimization problems, where the cost functions

defined on $\mathcal{T}$-spaces are locally Lipschitz. Locally Lipschitz functions are nonsmooth on a set of Lebesgue measure zero, but admit a generalized gradient at non-differentiable points. The field of nonsmooth optimization provides techniques like the subgradient method to minimize this class of nonsmooth functions.

As case studies, we considered the problem of computing a sample mean and central clustering in the domain of graphs. The cost functions are locally Lipschitz, but computation of a subgradient is computationally intractable. To cope with the computational complexity, we suggested an approximate subgradient method that chooses the opposite of a direction close to the generalized gradient as descent direction. We illustrated that the proposed method is capable to minimize the cost function of the case study. Even so the high computational complexity of deriving a subgradient demands a reevaluation of existing nonsmooth optimization methods and asks for devising algorithms that use approximations of the generalized gradient.

Appendix A. Introduction to $\mathcal{T}$-spaces

This section formally introduces $\mathcal{T}-spaces$ and presents proofs.

A.1 $\mathcal{T}$-spaces

Let $\mathcal{X} = \mathbb{R}^n$ be the n-dimensional Euclidean vector space, and let $\mathcal{T}$ be a subgroup of the group of all $n \times n$ permutation matrices. Then the binary operation

$$\cdot : \mathcal{T} \times \mathcal{X} \to \mathcal{X}, \quad (T,x) \mapsto Tx$$

is a group action of $\mathcal{T}$ on $\mathcal{X}$. For $x \in \mathcal{X}$, the *orbit* of x is denoted by

$$[x]_{\mathcal{T}} = \{Tx : T \in \mathcal{T}\}.$$

If no misunderstanding can occur, we write $[x]$ instead of $[x]_{\mathcal{T}}$.

A *$\mathcal{T}$-space* over $\mathcal{X}$ is the orbit space $\mathcal{X}_{\mathcal{T}} = \mathcal{X}/\mathcal{T}$ of all orbits of $x \in \mathcal{X}$ under the action of $\mathcal{T}$. We call $\mathcal{X}$ the *representation space* of $\mathcal{X}_{\mathcal{T}}$. By

$$\mu : \mathcal{X} \to \mathcal{X}_{\mathcal{T}}$$

we denote the *membership function* that sends vector representations to the structure they describe.

The following notations, definitions, and results are useful to simplify technicalities. We use capital letters $X, Y, Z, \ldots$ to denote the elements of $\mathcal{X}_{\mathcal{T}}$. Suppose that $X = \mu(x)$ for some $x \in \mathcal{X}$. Then we identify X with $[x]$ and make use of sloppy notations like, for example, $x \in X$ to denote $x \in [x]$.[10]

By $0_{\mathcal{T}}$ we denote the $\mathcal{T}$-zero of $\mathcal{X}_{\mathcal{T}}$. It is easy to show that $0_{\mathcal{T}}$ has only the zero element $0 \in \mathcal{X}$ as its unique representation vector.

Proposition 2 *Let $\mathcal{X}_{\mathcal{T}}$ be a $\mathcal{T}$-space over the metric vector space $\mathcal{X}$. Then $\mu^{-1}(0_{\mathcal{T}}) = [0] = \{0\}$.*

Proof Follows directly from the fact that each $T \in \mathcal{T}$ is homogeneous and injective. ■

A $\mathcal{T}$-space $\mathcal{X}_{\mathcal{T}}$ has, in fact, a $\mathcal{T}$-zero element, but it is unclear how to define an addition $+$ on $\mathcal{X}_{\mathcal{T}}$ such that $\mathcal{X}_{\mathcal{T}}$ together with $+$ forms a group. The absence of an additive group structure is one of the

10. The notation is sloppy, because X is an element in $\mathcal{X}_{\mathcal{T}}$ and not a set, whereas $[x]$ is a set of equivalent elements from $\mathcal{X}$.

major reasons why analytical tools for structured data are extremely rare compared to the plethora of powerful tools developed for feature vectors residing in some Banach space. To mitigate this drawback of $\mathcal{T}$-spaces $\mathcal{X}_\mathcal{T}$, we exploit the vector space axioms of $\mathcal{X}$ via the membership function μ.

We say a membership function $\mu : \mathcal{X} \to \mathcal{X}_\mathcal{T}$ is *$\mathcal{T}$-linear* if

$$\text{(TL1)} \quad [x+y] \subseteq [x] \oplus [y] = \{x' + y' : x' \in [x], y' \in [y]\},$$

$$\text{(TL2)} \quad [\lambda x] = \lambda [x] = \{\lambda x' : x' \in [x]\}$$

for all $x, y \in \mathcal{X}$, and for all $\lambda \in \mathbb{R}$. Note that for (TL1) we have a subset relation and for (TL2) equality. This definition has a sound notation but appears to be independent on μ at first glance. Since we identify the orbits $[x]$ in $\mathcal{X}$ with the elements $\mu(x)$ of $\mathcal{X}_\mathcal{T}$, we can rewrite (TL1) and (TL2) by slight abuse of notation

$$\begin{aligned} \mu(x+y) &\subseteq \mu(x) \oplus \mu(y), \\ \mu(\lambda x) &= \lambda \mu(x). \end{aligned}$$

Membership functions that are $\mathcal{T}$-linear preserve enough structure to transfer some geometrical and analytical concepts from $\mathcal{X}$ to $\mathcal{X}_\mathcal{T}$.

In contrast to the standard definition of linearity, we only require a subset relation in (TL1) rather than equality. The proof of Prop. 3 explains this issue.

Proposition 3 *Let $\mathcal{X}_\mathcal{T}$ be a $\mathcal{T}$-space over the Euclidean space $\mathcal{X}$.Then the membership function $\mu : \mathcal{X} \to \mathcal{X}_\mathcal{T}$ is T-linear.*

Proof Let $z = x + y$, and let $z' \in [x+y]$. Then there is an element $T \in \mathcal{T}$ with $z' = Tz$. Since T is linear by assumption, we obtain

$$z' = Tz = T(x+y) = Tx + Ty \in [x] \oplus [y].$$

This proves (TL1). The proof of (TL2) is similar. ■

For an intuitive understanding it is sometimes more convenient to use the following notation

$$\begin{aligned} \lambda X &= \mu(\lambda x), \\ X_x + Y_y &= \mu(x+y). \end{aligned}$$

It is important to note that the '+' symbol in $X_x + Y_y$ does not refer to some kind of addition in $\mathcal{X}_\mathcal{T}$. The notation $X_x + Y_y$ is simply an alternative and for our purposes more convenient way to refer to the element $\mu(x+y) \in \mathcal{X}_\mathcal{T}$.

We conclude this section with some further useful technical notations and results. Let $\mathcal{X}^n$ be the n-ary cartesian product of a metric vector space $\mathcal{X}$. Any real-valued function $f : \mathcal{X}^n \to \mathbb{R}$ induces functions

$$\begin{aligned} F^* &: \mathcal{X}_\mathcal{T}^n \to \mathbb{R}, \quad (X_1, \ldots, X_n) \mapsto \max\{f(x_1, \ldots, x_n) : x_i \in X_i\}, \\ F_* &: \mathcal{X}_\mathcal{T}^n \to \mathbb{R}, \quad (X_1, \ldots, X_n) \mapsto \min\{f(x_1, \ldots, x_n) : x_i \in X_i\}. \end{aligned}$$

Since $\mathcal{T}$ is finite, the orbits $[x]$ of x are finite. Hence, F^* and F_* assume an extremal value. We call F^* *maximizer* and F_* *minimizer* of f on $\mathcal{X}_\mathcal{T}^n$. Let F be either a maximizer or minimizer of f. The *support*

$$\operatorname{supp}(F | X_1, \ldots, X_n)$$

of F at $(X_1,\ldots,X_n) \in \mathcal{X}_{\mathcal{T}}^n$ is the set of all elements $(x_1,\ldots,x_n) \in \prod_{i=1}^n X_i$ with $f(x_1\ldots,x_n) = F(X_1,\ldots,X_n)$. The next results shows a useful property of the support.

Proposition 4 *Let F be the minimizer or maximizer of a function $f : \mathcal{X}^n \to \mathbb{R}$. Let $X_1,\ldots,X_n \in \mathcal{X}_{\mathcal{T}}$. Then for each $x_i \in X_i$ there are a $x_j \in X_j$ for all $j \in \{1,\ldots,n\} \setminus \{i\}$ such that $(x_1,\ldots,x_n) \in \operatorname{supp}(F|X_1,\ldots,X_n)$.*

Proof Let $x_i \in X_i$. Suppose that $(x_1^*,\ldots,x_n^*) \in \operatorname{supp}(F|X_1,\ldots,X_n)$. Then there is a transformation $T \in \mathcal{T}$ with $Tx_i = x_i^*$. Since $\mathcal{T}$ is a group, the inverse T^{-1} exists. Hence, $x_j = T^{-1}x_j^*$ is an element of X_j for all $j \neq i$. We have

$$F(X_1,\ldots,X_n) = f(x_1^*,\ldots,x_n^*) = f(Tx_1,\ldots,Tx_n) = f(x_1,\ldots,x_n).$$

This shows the assertion. ■

A.2 Metric $\mathcal{T}$-Spaces

Let $\mathcal{X}_{\mathcal{T}}$ be a $\mathcal{T}$-space over the metric vector space $(\mathcal{X},d)$. The minimizer

$$D_* : \mathcal{X}_{\mathcal{T}} \times \mathcal{X}_{\mathcal{T}} \to \mathbb{R}, \quad (X,Y) \mapsto \min\{d(x,y) : x \in X, y \in Y\}.$$

of the metric d is a distance measure on $\mathcal{X}_{\mathcal{T}}$. Theorem 5 shows that D_* is a metric.

Theorem 5 *Let $\mathcal{X}_{\mathcal{T}}$ be a $\mathcal{T}$-space over the metric space $(\mathcal{X},d)$. Then the minimizer D_* of d is a metric on $\mathcal{X}_{\mathcal{T}}$.*

Proof Let $X,Y,Z \in \mathcal{X}_{\mathcal{T}}$.

1. We show $D_*(X,Y) = 0 \Leftrightarrow X = Y$. Let $x \in X$ be a representation vector of X. According to Prop. 4 there is a $y \in Y$ such that $(x,y) \in \operatorname{supp}(D_*|x,y)$. We have

$$\begin{aligned} D_*(X,Y) = 0 &\Leftrightarrow \forall x \in X\, \exists y \in Y\; d(x,y) = 0 \\ &\Leftrightarrow \forall x \in X\, \exists y \in Y\; x = y \\ &\Leftrightarrow X = Y. \end{aligned}$$

2. Symmetry $D_*(X,Y) = D_*(Y,X)$ follows from symmetry of d.

3. We show $D_*(X,Z) \leq D_*(X,Y) + D_*(Y,Z)$. Let $(x,y) \in \operatorname{supp}(D_*|X,Y)$. There is a $z \in Z$ such that $(y,z) \in \operatorname{supp}(D_*|Y,Z)$. Then

$$\begin{aligned} D_*(X,Y) + D_*(Y,Z) &= d(x,y) + d(y,z) \\ &\geq d(x,z) \\ &\geq \min\{d(x,z) : x \in X, z \in Z\} \\ &= D_*(X,Z). \end{aligned}$$

■

Given the assumptions of Theorem 5, we call $(\mathcal{X}_{\mathcal{T}},D_*)$ *metric $\mathcal{T}$-space* over $(\mathcal{X},d)$. A metric space $\mathcal{X}$ is complete if every Cauchy sequence of points in $\mathcal{X}$ converges to a point from $\mathcal{X}$. The next result states that $\mathcal{X}_{\mathcal{T}}$ is complete if $\mathcal{X}$ is complete.

Theorem 6 *Any $\mathcal{T}$-space over a complete metric vector space is a complete metric space.*

Proof Let $\mathcal{X}_\mathcal{T}$ be a $\mathcal{T}$-space over the complete metric space $(\mathcal{X}, d)$. According to Theorem 5, $\mathcal{X}_\mathcal{T}$ is a metric space with metric D_*. To show that $\mathcal{X}_\mathcal{T}$ is complete, consider an arbitrary Cauchy sequence $(X_i)_{i\in\mathbb{N}}$ in $\mathcal{X}_\mathcal{T}$. We construct a Cauchy sequence (x_k) such that $(\mu(x_k))$ is a subsequence of (X_i). For any $k > 0$ there is a n_k such that $D_*(X_i, X_j) < 1/2^k$ for all $i, j > n_k$. For each k, there are $x_k \in X_{n_k}$ and $x_{k+1} \in X_{n_{k+1}}$ with $d(x_k, x_{k+1}) \leq 1/2^k$. By the triangle inequality, we have

$$d(x_i, x_j) \leq \sum_{k=i}^{j-1} d(x_k, x_{k+1}) \leq \frac{1}{2^{i-1}}$$

for any i, j with $i < j$. Hence, (x_k) is a Cauchy sequence in $\mathcal{X}$ and $(\mu(x_k))$ a subsequence of (X_i). Since $\mathcal{X}$ is complete, (x_k) converges to a limit point $x \in \mathcal{X}$. By continuity of μ, we have $\lim_{k\to\infty} \mu(x_k) = \mu(x)$, where $\mu(x) \in \mathcal{X}_\mathcal{T}$. Thus, the whole sequence (X_i) converges to $\mu(x)$. This shows that $\mathcal{X}_\mathcal{T}$ is complete. ■

A.3 $\mathcal{T}$-Spaces over Normed Vector Spaces

Let $\mathcal{X}_\mathcal{T}$ be a $\mathcal{T}$-space over the normed vector space $(\mathcal{X}, \|\cdot\|)$. As a normed vector space, $\mathcal{X}$ is a metric space with metric $d(x, y) = \|x - y\|$ for all $x, y \in \mathcal{X}$. For any $T \in \mathcal{T}$, we have

$$\|Tx\| = \|Tx - 0\| = \|Tx - T0\| = \|x - 0\| = \|x\|.$$

Hence, the minimizer $\|\cdot\|_*$ and maximizer $\|\cdot\|^*$ of $\|\cdot\|$ coincide, that is

$$\|X\|_* = \|X\|^* = \|x\| \tag{2}$$

for all $X \in \mathcal{X}_\mathcal{T}$ and for all $x \in X$.

We call the minimizer $\|\cdot\|_*$ the *$\mathcal{T}$-norm* induced by the norm $\|\cdot\|$. Note that a $\mathcal{T}$-norm is *not* a norm, because a $\mathcal{T}$-space has no well defined addition. But we can show that a $\mathcal{T}$-norm has norm-like properties. We use the notations λX for $\mu(\lambda X)$ and $X_x + Y_y$ for $\mu(x + y)$ as introduced in Section 3.2, p. 2696.

Proposition 7 *Let $(\mathcal{X}_\mathcal{T}, \|\cdot\|_*)$ be a $\mathcal{T}$-space over the normed space $(\mathcal{X}, \|\cdot\|)$. For all $X, Y \in \mathcal{X}_\mathcal{T}$, we have*

1. $\|X\|_* = 0$ *if, and only if,* $X = 0_\mathcal{T}$.
2. $\|\lambda X\|_* = |\lambda| \, \|X\|_*$ *for all* $\lambda \in \mathbb{R}$.
3. $\|X_x + Y_y\|_* \leq \|X\|_* + \|Y\|_*$ *for all* $x \in X$, $y \in Y$.

Proof Follows directly from first applying Equation (2) and then using the properties of the norm $\|\cdot\|$ defined on $\mathcal{X}$. ■

A.4 $\mathcal{T}$-Spaces over Inner Product Spaces

Let $\mathcal{X}_\mathcal{T}$ be a $\mathcal{T}$-space over the inner product space $(\mathcal{X}, \langle\cdot,\cdot\rangle)$, and let

$$\langle\cdot,\cdot\rangle^* : \mathcal{X}_\mathcal{T} \times \mathcal{X}_\mathcal{T} \to \mathbb{R}, \quad (X,Y) \mapsto \max\{\langle x,y\rangle : x \in X, y \in Y\}$$

be the maximizer of the inner product $\langle\cdot,\cdot\rangle$.

We call $\langle\cdot,\cdot\rangle^*$ *inner $\mathcal{T}$-product* induced by the inner product $\langle\cdot,\cdot\rangle$. The inner $\mathcal{T}$-product is *not* an inner product, because the maximum-operator in the definition of $\langle\cdot,\cdot\rangle^*$ does not preserve the bilinearity property of an inner product. But we shall show later that an inner $\mathcal{T}$-product satisfies some weaker properties of an inner product.

Any inner product space $\mathcal{X}$ is a normed space with norm $\|x\| = \sqrt{\langle x,x\rangle}$. The norm $\|\cdot\|$ on $\mathcal{X}$ in turn gives rise to the $\mathcal{T}$-norm $\|\cdot\|_*$ on $\mathcal{X}_\mathcal{T}$. The next result shows that a $\mathcal{T}$-norm is related to an inner $\mathcal{T}$-product in the same way as a norm to an inner product.

Proposition 8 *Let $(\mathcal{X}_\mathcal{T}, \langle\cdot,\cdot\rangle^*)$ be a $\mathcal{T}$-space over the inner product space $(\mathcal{X}, \langle\cdot,\cdot\rangle)$, and let $X \in \mathcal{X}_\mathcal{T}$. Then*

1. $\langle X,X\rangle^* = \langle x,x\rangle$ *for all* $x \in X$.
2. $\|X\|_* = \sqrt{\langle X,X\rangle^*}$.

Proof

1. X is the orbit of x under the group action $\mathcal{T}$. The assertion follows from the fact that each transformation T of $\mathcal{T}$ satisfies

   $$\langle Tx,Tx\rangle = \|Tx\|^2 = \|x\|^2 = \langle x,x\rangle$$

   for all $x \in \mathcal{X}$.

2. Follows from the first part by taking the square root.

■

Using Prop. 8 the following operations to construct $\|\cdot\|_*$ commute

$$\begin{array}{ccc} (\mathcal{X}, \langle\cdot,\cdot\rangle) & \xrightarrow{\langle\cdot,\cdot\rangle^*} & (\mathcal{X}_\mathcal{T}, \langle\cdot,\cdot\rangle^*) \\ {\scriptstyle \|\cdot\|=\sqrt{\langle\cdot,\cdot\rangle}}\downarrow & & \downarrow{\scriptstyle \|\cdot\|_*=\sqrt{\langle\cdot,\cdot\rangle^*}} \\ (\mathcal{X}, \|\cdot\|) & \xrightarrow[\|\cdot\|_*]{} & (\mathcal{X}_\mathcal{T}, \|\cdot\|_*)\,. \end{array}$$

Next we show that an inner $\mathcal{T}$-product satisfies some weaker properties related to an inner product.

Proposition 9 *Let $X,Y,Z \in \mathcal{X}_\mathcal{T}$, and let $x \in X$, $y \in Y$. Then*

1. $\langle X,X\rangle^* \geq 0$ *with* $\langle X,X\rangle^* = 0 \Leftrightarrow X = 0_\mathcal{T}$ (*positive definite*)
2. $\langle X,Y\rangle^* = \langle Y,X\rangle^*$ (*symmetric*)

3. $\langle \lambda X, Y \rangle^* = \lambda \langle X, Y \rangle^*$ *for* $\lambda \geq 0$ (*positive homogeneous*)

4. $\langle X_x + Y_y, Z \rangle^* \leq \langle X, Z \rangle^* + \langle Y, Z \rangle^*$ (*sublinear*)

Proof

1. Follows from Prop. 8 and the positive definiteness of $\langle \cdot, \cdot \rangle$.

2. Follows from the symmetry of $\langle \cdot, \cdot \rangle$.

3. Let $\lambda \geq 0$. Then

$$\begin{aligned} \langle \lambda X, Y \rangle^* &= \max \{ \langle \lambda x, y \rangle : x \in X, y \in Y \} \\ &= \lambda \max \{ \langle x, y \rangle : x \in X, y \in Y \} \\ &= \lambda \langle X, Y \rangle^* . \end{aligned}$$

4. Let $W = X_x + Y_y$, and let $(w,z) \in \operatorname{supp}(\langle \cdot, \cdot \rangle^* | W, Z)$. Since μ is $\mathcal{T}$-linear by Prop. 3, we have $W \subseteq X \oplus Y$. Hence, there are $x \in X$ and $y \in Y$ such that $w = x + y$. Thus,

$$\langle W, Z \rangle = \langle w, z \rangle = \langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle .$$

From $\langle x, z \rangle \leq \langle X, Z \rangle^*$ and $\langle y, z \rangle \leq \langle Y, Z \rangle^*$ follows the assertion.

■

From the proof of Prop. 9 follows that $\mathcal{T}$-linearity of the membership function partially preserves the structure of $\mathcal{X}$ such that the inner $\mathcal{T}$-product is a positive definite, symmetric, and sublinear in both arguments.

Any inner product space $(\mathcal{X}, \langle \cdot, \cdot \rangle)$ is a metric space with metric $d(x,y) = \|x - y\|$. The metric d induces a metric D_* on $\mathcal{X}_\mathcal{T}$. As for the $\mathcal{T}$-norm, we want to express D_* in terms of $\langle \cdot, \cdot \rangle^*$.

Proposition 10 *Let* $(\mathcal{X}_\mathcal{T}, \langle \cdot, \cdot \rangle^*)$ *be a* $\mathcal{T}$*-space over the inner product space* $(\mathcal{X}, \langle \cdot, \cdot \rangle)$. *Then for all* $X, Y \in \mathcal{X}_\mathcal{T}$, *we have*

$$D_*(X,Y)^2 = \|X\|_*^2 - 2 \langle X, Y \rangle^* + \|Y\|_*^2 .$$

Proof We have

$$\begin{aligned} D_*(X,Y)^2 &= \min \left\{ \|x - y\|^2 : x \in X, y \in Y \right\} \\ &= \min \{ \langle x - y, x - y \rangle : x \in X, y \in Y \} \\ &= \min \left\{ \|x\|^2 - 2 \langle x, y \rangle + \|y\|^2 : x \in X, y \in Y \right\} \\ &= \|x\|^2 - 2 \max \{ \langle x, y \rangle : x \in X, y \in Y \} + \|y\|^2 \\ &= \|X\|_*^2 - 2 \langle X, Y \rangle^* + \|Y\|_*^2 . \end{aligned}$$

■

Next, we extend the Cauchy-Schwarz inequality.

Theorem 11 *Let $(\mathcal{X}_\mathcal{T}, \langle\cdot,\cdot\rangle^*)$ be a $\mathcal{T}$-space over the inner product space $(\mathcal{X}, \langle\cdot,\cdot\rangle)$. Then*

$$\left|\langle X,Y\rangle^*\right| \leq \|X\|_* \|Y\|_*.$$

for all $X,Y \in \mathcal{X}_\mathcal{T}$

Proof Let $(x,y) \in \operatorname{supp}(\langle\cdot,\cdot\rangle^* | X,Y)$. Applying the conventional Cauchy-Schwarz inequality for vectors and using Eq. (2) yields

$$\left|\langle X,Y\rangle^*\right| = |\langle x,y\rangle| \leq \|x\| \|y\| = \|X\|_* \|Y\|_*.$$

■

Using Theorem 11, we can show that the angle of structures has a geometrical meaning. For two nonzero structures X and Y, the angle $\theta \in [0,\pi]$ between X and Y is defined (indirectly in terms of its cosine) by

$$\cos\theta = \frac{\langle X,Y\rangle^*}{\|X\|_* \|Y\|_*}.$$

Theorem 11 implies that

$$-1 \leq \frac{\langle X,Y\rangle^*}{\|X\|_* \|Y\|_*} \leq 1$$

and thus assures that this angle is well-defined. It is worthy to mention that $\langle X,Y\rangle^*$ has the same geometrical properties as an inner product, although it does not satisfy the algebraic properties of an inner product. Having the concept of an angle for structures, we can define structural orthogonality in the usual way. Two nonzero structures X and Y, written as $X \perp Y$, are *structurally orthogonal if* $\langle X,Y\rangle^* = 0$. Thus, Theorem 11 constitutes the starting point of a geometry of $\mathcal{T}$-spaces, which we do not further expand.

Appendix B. $\mathcal{T}$-Mappings

This section studies differential and the local Lipschitz property of mappings on $\mathcal{X}_\mathcal{T}$. The key result of this section is that the concept of gradient and its generalizations from nonsmooth analysis can be transferred in a well-defined manner to mappings on $\mathcal{T}$-spaces. Basic definitions and results on nonsmooth analysis of locally Lipschitz mappings are given in Appendix C.

B.1 Properties of $\mathcal{T}$-Mappings

Let $\mathcal{X}_\mathcal{T}$ be a $\mathcal{T}$-space over $\mathcal{X}$ and let $\mathcal{Y}$ be a set. A *$\mathcal{T}$-mapping* is a mapping of the form $F : \mathcal{X}_\mathcal{T} \to \mathcal{Y}$. If $\mathcal{Y}$ is a subset of $\mathbb{R}$, we also call F a *$\mathcal{T}$-function*.

Instead of studying a $\mathcal{T}$-mapping F directly, it is more convenient to consider its *representation mapping* defined by

$$f : \mathcal{X} \to \mathcal{Y}, \quad x \mapsto F \circ \mu(x).$$

Thus, we have to show that analyzing $\mathcal{T}$-mappings is equivalent to analyzing their representation mappings. For this we introduce the notion of $\mathcal{T}$-invariant mapping. A *$\mathcal{T}$-invariant mapping* is a mapping $f : \mathcal{X} \to \mathcal{Y}$ that is constant on the orbits $[x]_\mathcal{T}$ for all $x \in \mathcal{X}$. Obviously, representation mappings are $\mathcal{T}$-invariant. In addition, we have the following universal properties:[11]

11. A universal property can be regarded as some abstract property which requires the existence of a unique mapping under certain conditions.

(UP1) Each mapping $F : \mathcal{X}_\mathcal{T} \to \mathcal{Y}$ has a unique representation $f : \mathcal{X} \to \mathcal{Y}$ with $f = F \circ \mu$.

(UP2) Each $\mathcal{T}$-invariant mapping $f : \mathcal{X} \to \mathcal{Y}$ can be lifted in the obvious way to a unique mapping $F : \mathcal{X}_\mathcal{T} \to \mathcal{Y}$ with $f = F \circ \mu$.

Hence, by (UP1) and (UP2) we may safely identify $\mathcal{T}$-mappings with their representation mappings.

Next, we show that $\mathcal{T}$-spaces over complete metric vector spaces are universal quotients with respect to continuity, the Lipschitz property, and the local Lipschitz property. For this, we need some additional results.

Proposition 12 *Let $(\mathcal{X}_\mathcal{T}, D_*)$ be a metric $\mathcal{T}$-space over the metric space $(\mathcal{X}, d)$. Then the membership function $\mu : \mathcal{X} \to \mathcal{X}_\mathcal{T}$ is a continuous map.*

Proof Let $(x_i)_{i\in\mathbb{N}}$ be a sequence in $\mathcal{X}$ which converges to $x \in \mathcal{X}$. Let (X_i) be the sequence in $\mathcal{X}_\mathcal{T}$ with $X_i = \mu(x_i)$ for all $i \in \mathbb{N}$, and let $X = \mu(x)$. For any $\varepsilon > 0$ there is a number $n = n(\varepsilon)$ with $D_*(X_i, X) \leq d(x_i, x) < \varepsilon$ for all $i > n$. Hence, μ is continuous. ■

A mapping $f : \mathcal{X} \to \mathcal{Y}$ between topological spaces is open if for any open set $\mathcal{U} \in \mathcal{X}$, the image $f(\mathcal{U})$ is open in $\mathcal{Y}$.

Proposition 13 *Let $(\mathcal{X}_\mathcal{T}, D_*)$ be a metric $\mathcal{T}$-space over the metric vector space $(\mathcal{X}, d)$. Then $\mu : \mathcal{X} \to \mathcal{X}_\mathcal{T}$ is an open mapping.*

Proof It is sufficient to show that for any $x \in \mathcal{X}$ and any open neighborhood $\mathcal{U}$ of x there is an open neighborhood $\mathcal{V}_\mathcal{T}$ of $X = \mu(x)$ such that $\mathcal{V}_\mathcal{T} \subseteq \mu(\mathcal{U})$.

Let $x \in \mathcal{X}$, and let $\mathcal{U} \subseteq \mathcal{X}$ be an open set with $x \in \mathcal{U}$. Then there is $\varepsilon > 0$ such that the open neighborhood $\mathcal{N}(x, \varepsilon)$ is contained in $\mathcal{U}$. Let $\mathcal{U}_\mathcal{T} = \mu(U)$ and $\mathcal{V}_\mathcal{T} = \mu(\mathcal{N}(x, \varepsilon))$. Clearly, $X = \mu(x) \in \mathcal{V}_\mathcal{T} \subseteq \mathcal{U}_\mathcal{T}$. We show that $\mathcal{V}_\mathcal{T}$ is open. From $D_*(X, \mu(y)) \leq d(x, y) < \varepsilon$ for all $y \in \mathcal{N}(x, \varepsilon)$ follows $\mathcal{V}_\mathcal{T} \subseteq \mathcal{N}_\mathcal{T}(X, \varepsilon)$. Now let $Y \in \mathcal{N}_\mathcal{T}(X, \varepsilon)$. For x, we can find a a $y \in \mathcal{X}$ with $D_*(X, Y) = d(x, y) < \varepsilon$ by Prop. 4. Hence, $y \in \mathcal{N}(x, \varepsilon)$. This proves that $\mathcal{N}_\mathcal{T}(X, \varepsilon) = \mathcal{V}_\mathcal{T} \subseteq \mathcal{U}_\mathcal{T}$. ■

The next result shows the aforementioned universal property of $\mathcal{X}_\mathcal{T}$ with respect to continuity, the Lipschitz property, and the local Lipschitz property.

Proposition 14 *Let $\mathcal{X}, \mathcal{Y}$ be complete metric vector spaces, and let $\mathcal{X}_\mathcal{T}$ be a metric $\mathcal{T}$-space over X. Suppose that $f : \mathcal{X} \to \mathcal{Y}$ is a $\mathcal{T}$-invariant mapping. If f is continuous (Lipschitz, locally Lipschitz), then f lifts to a unique continuous (Lipschitz, locally Lipschitz) mapping $F : \mathcal{X}_\mathcal{T} \to \mathcal{Y}$ with $f(x) = F(\mu(x))$ for all $x \in \mathcal{X}$.*

Proof By (UP2), the existence of such a $\mathcal{T}$-mapping F implies uniqueness. Thus, it remains to show that F preserves continuity and both Lipschitz properties. In the following let $d_\mathcal{X}$ and $d_\mathcal{Y}$ denote the metric of $\mathcal{X}$ and $\mathcal{Y}$, resp., and let D_* be the metric of $\mathcal{X}_\mathcal{T}$ induced by $d_\mathcal{X}$.

Continuity. Let $\mathcal{U} \subseteq \mathcal{Y}$ open. Then $\mathcal{V} = f^{-1}(\mathcal{U})$ is open in $\mathcal{X}$, because f is continuous. From Prop. 13 follows that $\mathcal{V}_\mathcal{T} = \mu(\mathcal{V})$ is open in $\mathcal{X}_\mathcal{T}$. The assumption follows from the fact $\mathcal{V}_\mathcal{T} = F^{-1}(\mathcal{U})$.

Lipschitz property. Suppose there is a $L \geq 0$ such that

$$d_\mathcal{Y}(f(x), f(y)) \leq L \cdot d_\mathcal{X}(x, y)$$

for all $x,y \in \mathcal{X}$. Let $X,Y \in \mathcal{X}_{\mathcal{T}}$. For $(x,y) \in \operatorname{supp}(D_*|X,Y)$ we have

$$d_{\mathcal{Y}}(F(X),F(Y)) = d_{\mathcal{Y}}(f(x),f(y)) \leq L \cdot d_{\mathcal{X}}(x,y) = L \cdot D_*(X,Y).$$

Since X and Y were chosen arbitrarily, F is Lipschitz.
Local Lipschitz property. Let $X_0 \in \mathcal{X}_{\mathcal{T}}$, and let $x_0 \in X$. Since f is locally Lipschitz at x_0, there is a $L \geq 0$ such that

$$d_{\mathcal{Y}}(f(x),f(y)) \leq L \cdot d_{\mathcal{X}}(x,y)$$

for all x,y from some neighborhood $\mathcal{U} = \mathcal{N}(x_0,\varepsilon)$ in $\mathcal{X}$. Since $\mu : \mathcal{X} \to \mathcal{X}_{\mathcal{T}}$ is an open mapping, there is an open neighborhood $\mathcal{V}_{\mathcal{T}}$ of X_0 with $\mathcal{V}_{\mathcal{T}} \subseteq \mu(\mathcal{U})$. Let $X,Y \in \mathcal{V}_{\mathcal{T}}$ arbitrary. We choose $x,y,y' \in \mu^{-1}(\mathcal{V}_{\mathcal{T}}) \subseteq \mathcal{U}$ with $\mu(x) = X, \mu(y) = \mu(y') = Y$, and $(x,y) \in \operatorname{supp}(D_*|X,Y)$. From $x,y' \in \mathcal{U}$ follows $d_{\mathcal{X}}(x,y') < \varepsilon$ and from $(x,y) \in \operatorname{supp}(D_*|X,Y)$ follows $D_*(X,Y) = d_{\mathcal{X}}(x,y)$. Combining both relations and using that D_* is a minimizer of $d_{\mathcal{X}}$ yields

$$D_*(X,Y) = d_{\mathcal{X}}(x,y) \leq d_{\mathcal{X}}(x,y') < \varepsilon.$$

Hence, we have

$$d_{\mathcal{Y}}(F(X),F(Y)) = d_{\mathcal{Y}}(f(x),f(y)) \leq L \cdot d_{\mathcal{X}}(x,y) = L \cdot D_*(X,Y).$$

Since $X,Y \in \mathcal{V}_{\mathcal{T}}$ were chosen arbitrarily, F is locally Lipschitz at X_0. ■

Now we want to study differential properties of $\mathcal{T}$-mappings via their representation mappings. Let $\mathcal{X}_{\mathcal{T}}$ be a $\mathcal{T}$-space over the Euclidean space $(\mathcal{X},\|\cdot\|)$ and let $\mathcal{Y}$ be another Euclidean space. Suppose that $f : \mathcal{X} \to \mathcal{Y}$ is a $\mathcal{T}$-mapping, which is differentiable at $\bar{x} \in \mathcal{X}$. By

$$Df(\bar{x}) : \mathcal{X} \to \mathcal{Y}, \quad x \mapsto Df(\bar{x})(x)$$

we denote the derivative of f at $\bar{x}$.

Theorem 15 *Let $\mathcal{X}$ and $\mathcal{Y}$ be Euclidean spaces, and let $\mathcal{X}_{\mathcal{T}}$ be a $\mathcal{T}$-space over $\mathcal{X}$. Suppose that $f : \mathcal{X} \to \mathcal{Y}$ is a $\mathcal{T}$-mapping, which is differentiable at $\bar{x} \in \mathcal{X}$. Then f is differentiable at all points $x \in [\bar{x}]$. In addition, we have*

$$Df(T\bar{x}) = Df(\bar{x}) \circ T^{-1}$$

for all $T \in \mathcal{T}$.

Proof By $\|\cdot\|_{\mathcal{X}}$ and $\|\cdot\|_{\mathcal{Y}}$ we denote the norm defined on $\mathcal{X}$ and $\mathcal{Y}$, respectively. Let $x \in [\bar{x}]$. We show that f is differentiable in x. Let $T \in \mathcal{T}$ with $T\bar{x} = x$. For $h \neq 0$, we define the mapping

$$\begin{aligned} r(h) &= \frac{\left\| f(x+h) - f(x) - Df(\bar{x})\left(T^{-1}h\right)\right\|_{\mathcal{Y}}}{\|h\|_{\mathcal{X}}} \\ &= \frac{\left\| f(T\bar{x}+TT^{-1}h) - f(T\bar{x}) - Df(\bar{x})\left(T^{-1}h\right)\right\|_{\mathcal{Y}}}{\|TT^{-1}h\|_{\mathcal{X}}}. \end{aligned}$$

We set $h' = T^{-1}h$ and obtain

$$\begin{aligned} r(h) &= \frac{\|f(T\bar{x}+Th') - f(T\bar{x}) - Df(\bar{x})(h')\|_{\mathcal{Y}}}{\|Th'\|_{\mathcal{X}}} \\ &= \frac{\|f(T(\bar{x}+h')) - f(T\bar{x}) - Df(\bar{x})(h')\|_{\mathcal{Y}}}{\|Th'\|_{\mathcal{X}}}. \end{aligned}$$

Since f is $\mathcal{T}$-invariant, we have $f(T(\bar{x}+h')) = f(\bar{x}+h')$ and $f(T\bar{x}) = f(\bar{x})$. In addition, we have $\|Th'\| = \|h'\|$, because T is an isometry. Thus,

$$r(h) = \frac{\|f(\bar{x}+h') - f(\bar{x}) - Df(\bar{x})(h')\|_{\mathcal{Y}}}{\|h'\|_{\mathcal{X}}}.$$

Since f is differentiable in $\bar{x}$, we have

$$\lim_{h\to 0} r(h) = \lim_{h'\to 0} r(h) = 0.$$

Hence, f is differentiable at x with derivative $Df(x) = Df(T\bar{x}) = Df(\bar{x}) \circ T^{-1}$. ■

A $\mathcal{T}$-mapping $F : \mathcal{X}_{\mathcal{T}} \to \mathcal{Y}$ is said to be *$\mathcal{T}$-differentiable* at $\bar{X} \in \mathcal{X}_{\mathcal{T}}$ if its representation function is differentiable at an arbitrary vector representation of $\bar{X}$. From Theorem 15 follows that differentiability of f at an arbitrary vector representation of $\bar{X}$ implies differentiability at all vector representations of $\bar{X}$. Hence, $\mathcal{T}$-differentiability at $\bar{X} \in \mathcal{X}_{\mathcal{T}}$ is independent from the particular vector representation of $\bar{X}$ and therefore well-defined.

B.2 $\mathcal{T}$-Differentiable Functions

In this section, we study differential properties of $\mathcal{T}$-functions of the form $F : \mathcal{X}_{\mathcal{T}} \to \mathbb{R}$.

Let $f : \mathcal{X} \to \mathbb{R}$ be the representation function of F. Suppose that f is differentiable at $x \in \mathcal{X}$ with gradient $\nabla f(\bar{x})$. Then, by Theorem 15, the $\mathcal{T}$-function F is $\mathcal{T}$-differentiable at $\bar{X} = \mu(\bar{x})$. We call the structure

$$\nabla F(\bar{X}) = \mu(\nabla f(\bar{x})).$$

$\mathcal{T}$-gradient of F at $\bar{X}$. We show that the $\mathcal{T}$-gradient is well-defined, that is independent from the particular choice of vector representation $\bar{x} \in \bar{X}$. To see this let $T \in \mathcal{T}$ be an arbitrary orthogonal transformation from $\mathcal{T}$. By Theorem 15, we have

$$\nabla f(T\bar{x}) = \nabla f(\bar{x})T^{-1} = \nabla f(\bar{x})T',$$

where $T' = T^{-1}$ is the transpose of T. From $\nabla f(\bar{x})T' = T\nabla f(\bar{x})$ follows $\nabla f(T\bar{x}) = T\nabla f(\bar{x})$. Thus, we have

$$\mu(\nabla f(T\bar{x})) = \mu(T\nabla f(\bar{x})) = \mu(\nabla f(\bar{x}))$$

showing that the $\mathcal{T}$-gradient is well-defined.

The $\mathcal{T}$-gradient induces the $\mathcal{T}$-function

$$\nabla F(\bar{X}) : \mathcal{X}_{\mathcal{T}} \to \mathbb{R}, \quad X \mapsto \langle \nabla F(\bar{X}), X \rangle^*.$$

We use this function for showing that the geometrical properties of a gradient also hold for a $\mathcal{T}$-gradient. For this, we define the *directional $\mathcal{T}$-derivative* of F at $\bar{X}$ along the direction V with $\|V\|_* = 1$ by

$$D_V F(\bar{X}) = \langle V, \nabla F(\bar{X}) \rangle^*.$$

The first geometrical result shows that the $\mathcal{T}$-gradient is a structure pointing to the direction of steepest ascent.

Proposition 16 *Let $f : \mathcal{X} \to \mathbb{R}$ be a continuously differentiable representation function of a $\mathcal{T}$-function $F : \mathcal{X}_{\mathcal{T}} \to \mathbb{R}$. Then for all $\bar{X} \in \mathcal{X}_{\mathcal{T}} \setminus \{0_{\mathcal{T}}\}$ and all $V \in \mathcal{U}_{\mathcal{T}} = \{X \in \mathcal{X} : \|X\|_* = 1\}$, we have*

$$\frac{\nabla F(\bar{X})}{\|\nabla F(\bar{X})\|_*} = \arg\max_{V \in \mathcal{U}_{\mathcal{T}}} D_V F(\bar{X}).$$

Proof From the definition of the directional derivative and the implications of Theorem 11 follows

$$D_V F(\bar{X}) = \langle V, \nabla F(\bar{X}) \rangle^* = \|V\|_* \, \|\nabla F(\bar{X})\|_* \cos\alpha,$$

where α is the angle between V and $\nabla F(\bar{X})$. Hence, the directional derivative $D_V F(\bar{X})$ becomes maximal if V points to the same direction as $\nabla F(\bar{X})$. ■

Next, we show that the necessary condition for optimality can be transferred to $\mathcal{T}$-differentiable $\mathcal{T}$-functions.

Proposition 17 *Let $F : \mathcal{X}_{\mathcal{T}} \to \mathbb{R}$ be a $\mathcal{T}$-function with a partial differentiable representation function. If $X \in \mathcal{X}_{\mathcal{T}}$ is a local optimum of F, then we have*

$$\nabla F(X) = 0_{\mathcal{T}}.$$

Proof Let $f : \mathcal{X} \to \mathbb{R}$ be the representation function of F, and let $x \in X$ be a vector representation of X. Since f is partial differentiable, the gradient of f at x exists. By definition of the $\mathcal{T}$-gradient, we have

$$\mu(\nabla f(x)) = \nabla F(X) = 0_{\mathcal{T}}.$$

From Prop. 2 follows that 0 is the unique vector representation of $\nabla F(X)$. Thus, any vector representation x of X is a local optimum of the representation function f. Without loss of generality, we assume that $x \in X$ is a local minimum. Then there is an open neighborhood $\mathcal{U}$ of x with $f(x) \leq f(x')$ for all $x' \in \mathcal{U}$. Since μ is an open mapping by Prop. 13, the set $\mathcal{U}_{\mathcal{T}} = \mu(\mathcal{U})$ is an open neighborhood of X. From the $\mathcal{T}$-invariance of f follows that $F(X) = f(x) \leq f(x') = F(X')$ for all $X' \in \mathcal{U}_{\mathcal{T}}$. This shows that F is a local minimum. ■

An immediate consequence of the proof is that if x is a local minimum (maximum) of the representation function f then all $x' \in [x]$ are local minima (maxima).

B.3 Pointwise Maximizers

This section introduces and studies differential properties of pointwise maximizers and applies the results to structural similarity and distance functions.

B.3.1 POINTWISE MAXIMIZERS

The *pointwise maximizer* of functions $f_1, \ldots, f_m : \mathcal{U} \to \mathbb{R}$ defined on an open subset $\mathcal{U} \subseteq \mathbb{R}^n$ is the function $f : \mathcal{U} \to \mathbb{R}$ with

$$f(x) = \max_{1 \leq i \leq m} f_i(x).$$

We call the set $\operatorname{supp}(f) = \{f_i : 1 \leq i \leq m\}$ the *support* of f, and its elements *support functions*.

Theorem 18 *Let $f : \mathcal{U} \to \mathbb{R}$ be a pointwise maximizer with finite support* $\operatorname{supp}(f)$. *We have:*

- *If all* $f_i \in \text{supp}(f)$ *are locally Lipschitz at* x*, then* f *is locally Lipschitz at* x *and*

$$\partial f(x) \subseteq con\{\partial f_i(x) : f_i \in \text{supp}(f) \wedge f_i(x) = f(x)\}. \tag{3}$$

- *If all* $f_i \in \text{supp}(f)$ *are regular at* x*, then* f *is regular at* x *and equality in (3) holds.*

- *If all* $f_i \in \text{supp}(f)$ *are smooth at* x*, then* f *is regular at* x *and*

$$\partial f(x) = con\{\nabla f_i(x) : f_i \in \text{supp}(f) \wedge f_i(x) = f(x)\}.$$

Proof Mäkelä and Neittaanmäki (1992), Theorem 3.2.12 and Corollary 3.2.14. ■

If all support functions of f^* are locally Lipschitz, then f^* is also locally Lipschitz and admits a generalized gradient at any point from $\mathcal{U}$. In addition, f^* is differentiable almost everywhere on $\mathcal{U}$ by Rademacher's Theorem (see Appendix, Theorem 23).

Similarly, we can define in the obvious way the *pointwise minimizer* of a finite set of functions. According to Theorem 19, all statements made on pointwise maximizers carry over to pointwise minimizers.

Theorem 19 *If* f *be locally Lipschitz at* x*, then*

$$\partial f(\lambda x) = \lambda \partial f(x)$$

for all $\lambda \in \mathbb{R}$.

Proof Mäkelä and Neittaanmäki (1992), Theorem 3.2.4. ■

In the remainder of this section, we consider similarity and distance functions as examples of pointwise maximizers and minimizers, respectively.

B.3.2 SIMILARITY FUNCTIONS: THE GENERAL CASE

We consider similarity functions of the form

$$S^* : \mathcal{X}_\mathcal{T} \times \mathcal{X}_\mathcal{T} \to \mathbb{R}, \quad (X,Y) \mapsto \max_{x \in X, y \in Y} s(x,y)$$

that are maximizers of similarity functions $s : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. For a given $Y \in \mathcal{X}_\mathcal{T}$, we define the function

$$s_Y : \mathcal{X} \to \mathbb{R}, \quad x \mapsto S^*(\mu(x), Y).$$

The function s_Y represents $S^*(\cdot, Y)$ and is a pointwise maximizer with support

$$\text{supp}(s_Y) = \{s_y : s_y(\cdot) = s(\cdot, y), y \in Y\}.$$

If the support functions of s_Y are locally Lipschitz, regular, or smooth, we can apply Theorem 18 to show that s_Y is locally Lipschitz, admits a generalized gradient at each point of its domain, and is differentiable almost everywhere.

B.3.3 SIMILARITY FUNCTIONS: THE INNER $\mathcal{T}$-PRODUCT

As a specific example of a similarity function as a pointwise maximizer, we consider the inner $\mathcal{T}$-product. Suppose that $S^*(\cdot,\cdot) = \langle\cdot,\cdot\rangle^*$. For a fixed structure $Y \in \mathcal{X}_{\mathcal{T}}$, the support of the pointwise maximizer s_Y is of the form

$$\operatorname{supp}(s_Y) = \{s_y : s_y(\cdot) = \langle\cdot, y\rangle, y \in Y\},$$

As linear functions, these support functions s_y are continuously differentiable. From Theorem 18 follows

- s_Y is locally Lipschitz and regular,
- the generalized gradient $\partial s_Y(x)$ is the convex set

$$\partial s_Y(x) = \operatorname{con}\{y \in Y : (x,y) \in \operatorname{supp}(s_Y|x)\}.$$

The next statement follows directly from Prop. 9.

Proposition 20 *The function $s_Y : \mathcal{X} \to \mathbb{R}$ is convex.*

Proof From Prop. 9 follows that s_y is positively homogeneous and sublinear. Hence, s_Y is convex. ■

B.3.4 DISTANCE FUNCTIONS: THE GENERAL CASE

Suppose that we are given an arbitrary distance function of the form

$$D_* : \mathcal{X}_{\mathcal{T}} \times \mathcal{X}_{\mathcal{T}} \to \mathbb{R}, \quad (X,Y) \mapsto \min_{x\in X, y\in Y} d(x,y).$$

To apply the theorems on pointwise maximizers, we consider the function

$$\widehat{D}_*(X,Y) = -D_*(X,Y) = \max_{x\in X, y\in Y} -d(x,y).$$

For a given $Y \in \mathcal{X}_{\mathcal{T}}$, we define the function

$$\widehat{d_Y} : \mathcal{X} \to \mathbb{R}, \quad x \mapsto \widehat{D}_*(\mu(x), Y).$$

The function $\widehat{d_Y}$ represents $\widehat{D}_*(\cdot, Y)$ and is a pointwise maximizer with support

$$\operatorname{supp}\left(\widehat{d_Y}\right) = \left\{\widehat{d_y} : \widehat{d_y}(\cdot) = -d(\cdot, y), y \in Y\right\}.$$

B.3.5 DISTANCE FUNCTIONS: THE METRIC D_*

Let D_* be the metric on $\mathcal{X}_{\mathcal{T}}$ induced by a metric d on $\mathcal{X}$ of the form $d(x,y) = \|x-y\|$. For a given $Y \in \mathcal{X}_{\mathcal{T}}$, we define the function

$$\widehat{d_Y} : \mathcal{X} \to \mathbb{R}, \quad x \mapsto \widehat{D}_*(\mu(x), Y).$$

The function $\widehat{d_Y}$ represents $\widehat{D}_*(\cdot, Y)$ and is a pointwise maximizer with support

$$\operatorname{supp}\left(\widehat{d_Y}\right) = \{\widehat{d_y} : y \in Y\},$$

where $\widehat{d_y}(x) = -\|x-y\|$ for all $x \in \mathcal{X}$. The support functions of $\widehat{d_Y}$ are locally Lipschitz and regular on $\mathcal{X}$, and smooth on $\mathcal{X} \setminus \{y\}$. From Theorem 18 follows

- $\widehat{d_Y}$ is locally Lipschitz and regular,
- the generalized gradient $\partial\widehat{d_Y}(x)$ is the convex set

$$\partial\widehat{d_Y}(x) = \begin{cases} \operatorname{con}\left\{-\frac{(x-y)}{\|x-y\|} : y \in Y \wedge (y,x) \in \operatorname{supp}(\widehat{d_Y}|x)\right\} & : \quad x \neq y \\ \{y \in \mathcal{X}_\mathcal{T} : \|y\| \leq 1\} & : \quad x = y. \end{cases}$$

As in the Euclidean space, the squared structural Euclidean metric D_*^2 turns out to be more convenient than D_*. As opposed to $\widehat{d_Y}$, the support functions of the squared function $\widehat{d_Y^2}$ are continuously differentiable at all points from $\mathcal{X}$. In particular, we have

- $\widehat{d_Y^2}$ is locally Lipschitz and regular,
- the generalized gradient $\partial\widehat{d_Y^2}(x)$ is the convex set

$$\partial\widehat{d_Y^2}(x) = \operatorname{con}\left\{-2(x-y) : y \in Y \wedge (y,x) \in \operatorname{supp}\left(\widehat{d_Y^2}|x\right)\right\}.$$

Appendix C. Locally Lipschitz Functions

We review some basic properties of locally Lipschitz functions and their generalized gradients. Unless otherwise stated proofs can be found in Clarke (1990), Section 2.3. For a detailed treatment to the first-order generalized derivative we refer to Clarke (1990); Mäkelä and Neittaanmäki (1992).

Let $(\mathcal{X}, d_X)$ and $(\mathcal{Y}, d_Y)$ be metric spaces, and let $\mathcal{U} \subseteq \mathcal{X}$ be an open set. A map $f : \mathcal{X} \to \mathcal{Y}$ is *Lipschitz* on $\mathcal{U}$ if there is a scalar $L \geq 0$ with

$$d_Y(f(u), f(v)) \leq L \cdot d_X(u, v)$$

for all $u, v \in \mathcal{U}$. We say that f is *locally Lipschitz* at $u \in \mathcal{U}$ if f is Lipschitz on some ε-neighborhood $\mathcal{N}(u, \varepsilon) \subseteq \mathcal{U}$ of u.

Proposition 21 *Let $f, g : \mathcal{U} \subseteq \mathcal{X} \to \mathbb{R}$ be locally Lipschitz at u, and let $\lambda \in \mathbb{R}$ be a scalar. Then λf, $f + g$, and $f \cdot g$ are locally Lipschitz at u. If $g(u) \neq 0$, then f/g is locally Lipschitz at u.*

Proposition 22 *Let $f : \mathcal{X} \to \mathcal{Y}$ be locally Lipschitz at x, and let $g : \mathcal{X} \to \mathcal{Z}$ be locally Lipschitz at $y = f(x)$. Then $h = g \circ f$ is locally Lipschitz at x.*

Proof Let $N(x, \varepsilon_x) \subseteq \mathcal{X}, N(y, \varepsilon_y) \subseteq \mathcal{Y}$ be neighborhoods of x and y satisfying the following properties: (i) $f(N(x, \varepsilon_x)) \subseteq N(y, \varepsilon_y)$, (ii) there are $L_x, L_y \geq 0$ such that $d_Y(f(u), f(v)) \leq L_x \cdot d_X(u, v)$ for all $u, v \in N(x, \varepsilon_x)$ and $d_Z(g(p), g(q)) \leq L_y \cdot d_Y(p, q)$ for all $p, q \in N(y, \varepsilon_y)$. For any $u, v \in \mathcal{X}$, we have

$$d_Z(g \circ f(u), g \circ f(v)) \leq L_y d_Y(f(u), f(v)) \leq L_y L_x d_X(u, v).$$

■

Theorem 23 (Rademacher) *Let $\mathcal{U} \subseteq \mathbb{R}^n$ be a nonempty open set, and let $f : \mathcal{U} \to \mathbb{R}$ be locally Lipschitz. Then the set of points at which f is not differentiable has Lebesgue measure zero.*

The function f is *directionally differentiable* at $x \in \mathcal{U}$ if the limit

$$f'(x,d) = \lim_{t \downarrow 0} \frac{f(x+td) - f(x)}{t}$$

exists for all directions $d \in \mathbb{R}^n$. In this case, the value $f'(x,d)$ is the *directional derivative* of f at x in the direction d. We call the function f *directionally differentiable* if f is directionally differentiable at all points $x \in \mathcal{U}$.

The *generalized directional derivative* of f at $x \in \mathcal{U}$ in the direction $d \in \mathcal{X}$ is defined by

$$f^{\circ}(x,d) = \lim_{t \downarrow 0, y \to x} \sup \frac{f(y+td) - f(y)}{t},$$

where $t \downarrow 0$ and $y \to x$ are sequences such that $y + td$ is always in $\mathcal{U}$.

We say f is *regular* at $x \in \mathcal{U}$ if the following conditions are satisfied

1. f is directionally differentiable at x.
2. $f^{\circ}(\bar{x},d) = f'(x,d)$ for all $d \in \mathbb{R}^n$.

A function f is said to be *smooth* at x if f is continuously differentiable at x. We have the following implications.

Proposition 24 *Let $f : \mathcal{U} \to \mathbb{R}$ be a function. The following implications hold:*

1. *f is smooth at x, then f is locally Lipschitz, regular, continuous, and differentiable at x.*
2. *f is locally Lipschitz or differentiable at x, then f is continuous at x.*

Proof

1. Smoothness implies differentiability and continuity are well-known results from analysis. The locally Lipschitz property follows from Mäkelä and Neittaanmäki (1992), Lemma 3.1.6 and regularity from Mäkelä and Neittaanmäki (1992), Theorem 3.2.2.
2. Both assertions are again well-known results from analysis.

■

The *generalized gradient* $\partial f(x)$ of f at x is the set

$$\partial f(x) = \{y \in \mathcal{X} : f^{\circ}(x,d) \geq \langle y,d \rangle \text{ for all } d \in \mathcal{X}\}.$$

The elements of the set $\partial f(x)$ are called *subgradients* of f at x.

Theorem 25 *Let $f : \mathcal{U} \to \mathbb{R}$ be a function on the open subset $\mathcal{U}$. We have:*

- *If f is locally Lipschitz and differentiable at x, then*

$$\nabla f(x) \in \partial f(x).$$

- *If f is locally Lipschitz, regular, and differentiable at x, then*

$$\partial f(x) = \{\nabla f(x)\}.$$

Proof Mäkelä and Neittaanmäki (1992), Theorem 3.1.5, Theorem 3.1.7, and Theorem 3.2.4. ■

Appendix D. The Graduated Assignment Algorithm

We use the graduated assignment algorithm to approximate the NP-hard squared distance $D_*(X,Y)^2$ between two weighted graphs and to determine a subgradient from the generalized gradient ∂d_Y^2 (see Section B.3.5). According to Prop. 10, the squared distance $D_*(X,Y)^2$ can be expressed by the inner $\mathcal{T}$-product. Here, we determine $D_*(X,Y)^2$ via $\langle X,Y\rangle^*$.

Let X and Y be weighted graphs with weighted adjacency matrices $X=(x_{ij})$ and $Y=(y_{ij})$. Suppose that X and Y are of order n and m, respectively. Without loss of generality, we assume that $n<m$. By $\Pi=\Pi^{n\times m}$ we denote the set of $(n\times m)$-match matrices $M=(m_{ij})$ with elements from $[0,1]$ such that each row sums to 1 and each column sums to n/m. Then we have

$$\langle X,Y\rangle^* = \max_{M\in\Pi} \left\langle M'XM, Y\right\rangle,$$

where M' denotes the transpose of M. To compute the inner $\mathcal{T}$-product, graduated assignment minimizes

$$F(M) = -\frac{1}{2}\left\langle M'XM, Y\right\rangle.$$

subject to $M\in\Pi$. Suppose that M_0 is an optimal solution. Then $2\left(M_0'XM_0-Y\right)$ is a subgradient from the generalized gradient $\partial d_Y^2(X)$.

The core of the algorithm implements a deterministic annealing process with annealing parameter T by the following iteration scheme

$$m_{ij}^{(t+1)} = a_i b_j \exp\left(-\frac{1}{T}\sum_{r=1}^{n}\sum_{s=1}^{m} m_{rs}^{(t)}\left\langle x_{ir}, y_{js}\right\rangle\right),$$

where t denotes the time step. The scaling factors a_i, b_i computed by Sinkhorn's algorithm (Sinkhorn, 1964) enforce the constraints of the match matrix. The algorithm in detail is described in Gold, Rangarajan, and Mjolsness (1996).

References

B. Bonev, F. Escolano, M.A. Lozano, P. Suau, M.A. Cazorla, and W. Aguilar. Constellations and the unsupervised learning of graph. *Lecture Notes In Computer Science*, 4538:340, 2007.

A. Caprara and G. Lancia. Structural alignment of large–size proteins via lagrangian relaxation. In *Proceedings of the 6th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2002)*, pages 100–108, 2002.

A. Caprara, R. Carr, S. Istrail, G. Lancia, and B. Walenz. 1001 optimal pdb structure alignments: Integer programming methods for finding the maximum contact map overlap. *Journal of Computational Biology*, 11(1):27–52, 2004.

F. H. Clarke. *Optimization and Nonsmooth Analysis*. SIAM, 1990.

T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. CRC Press, 2000.

L. Dehaspe, H. Toivonen, and R.D. King. Finding frequent substructures in chemical compounds. In *In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 30–36, 1998.

P. Dosch and E. Valveny. Report on the second symbol recognition contest. In *GREC 2005*, pages 381–397, 2006.

M.A. Eshera and K.S. Fu. An image understanding system using attributed symbolic representation and inexact graph-matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:604–618, 1986.

T. Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1): 49–58, 2003.

T. Gärtner. *Kernels for Structured Data*. PhD thesis, University of Bonn, Germany, 2005.

S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388, 1996.

S. Gold, A. Rangarajan, and E. Mjolsness. Learning with pre-knowledge: Clustering with point and graph-matching distance measures. *Neural Computation*, 8:787–804, 1996.

L. Goldfarb. A new approach to pattern recognition. *Progress in Pattern Recognition*, 2:241–402, 1985.

D. Goldman, S. Istrail, and C.H. Papadimitriou. Algorithmic aspects of protein structure similarity. In *40th Annual Symposium on Foundations of Computer Science*, pages 512–521, 1999.

T. Graepel and K. Obermayer. A self-organizing map for proximity data. *Neural Computation*, 11: 139–155, 1999.

T. Graepel, R. Herbrich, P. Bollmann-Sdorra, and K. Obermayer. Classification on pairwise proximity data. In *Advances in Neural Information Processing*, volume 11, pages 438–444,, 1999.

S. Günter and H. Bunke. Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters*, 23:401–417, 2002.

D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *In Proceedings of the ACM SIGMOD International Conference on Management of Database*, pages 1–12, 2000.

M. Hein, O. Bousquet, and B. Schölkopf. Maximal margin classification for metric spaces. *Journal of Computer and System Sciences*, 71(3):333–359, 2005.

R. Herbrich, T. Graepel, P. Bollmann-Sdorra, and K. Obermayer. Learning a preference relation for information retrieval. In *Workshop Text Categorization and Machine Learning, International Conference on Machine Learning 1998*, page 8084, 1998.

D. Hochbaum and D. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

S. Hochreiter and K. Obermayer. *Kernel Methods in Computational Biology*, chapter Gene Selection for Microarray Data, pages 319–356. MIT Press, Cambridge, Massachusetts, 2004.

S. Hochreiter and K. Obermayer. Support vector machines for dyadic data. *Neural Computation*, 18:1472–1510, 2006.

T. Hofmann and J. M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(1):1–14, 1997.

L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, 233(1):123–38, 1993.

A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *In Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'00)*, pages 13–23, 2000.

B.J. Jain and M. Lappe. Joining softassign and dynamic programming for the contact map overlap problem. In *Proceedings of the 1st International Conference on Bioinformatics Research and Development (BIRD 2007)*, 2007.

B.J. Jain and K. Obermayer. On the sample mean of graphs. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2008)*, 2008.

B.J. Jain and K. Obermayer. Multiple alignment of contact maps. In *International Joint Conference on Neural Networks (IJCNN 2009)*, 2009.

B.J. Jain and F. Wysotzki. Central clustering of attributed graphs. *Machine Learning*, 56(1):169–207, 2004.

X. Jiang, A. Münger, and H. Bunke. On median graphs: Properties, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1144–1151, 2001.

H Kashima, K Tsuda, and A Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 2003.

J. Kazius, R. McGuire, and R. Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 48(1):312–320, 2005.

M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *In Proceedings of IEEE International Conference on Data Mining ICDM'01*, pages 313–320, 2001.

V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 10:707–710, 1966.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

M. A. Lozano and F. Escolano. Em algorithm for clustering an ensemble of graphs with comb matching. In A. Rangarajan, M. Figueiredo, and J. Zerubia, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition, EMMCVPR 2003*, LNCS 2683, pages 52–67. Springer, 2003.

B. Luo, R. C. Wilson, and E. R. Hancock. Spectral embedding of graphs. *Pattern Recognition*, 36 (10):2213–2230, 2003.

M.M. Mäkelä and P. Neittaanmäki. *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*. World Scientific, 1992.

H. Quang Minh and T. Hofmann. Learning over compact metric spaces. In *Proceedings of the 17th Annual Conference on Learning Theory (COLT 2004)*, 2004.

R. Myers, R.C. Wilson, and E.R. Hancock. Bayesian graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):628–635, 2000.

M. Neuhaus and H. Bunke. A graph matching based approach to fingerprint classification using directional variance. In *AVBPA 2005*, pages 191–200, 2005.

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

N. Ratliff, J. A. Bagnell, and M. Zinkevich. Subgradient methods for maximum margin structured learning. In *In Proceddings of the 23rd International Conference on Machine Learning (ICML'06), Workshop on Learning in Structured Output Spaces*, 2006.

J.W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structure. *Journal of Computer-Aided Molecular Design*, 16:521–533, 2002.

K. Riesen and H. Bunke. IAM graph database repository for graph based pattern recognition and machine learning. In *SSPR 2008*, pages 287–297, 2008.

A. Robles-Kelly and E.R. Hancock. Graph edit distance from spectral seriation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):365– 378, 2005.

A. Sanfeliu and K.S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:353–362, 1983.

I. J. Schoenberg. On certain metric spaces arising from euclidean spaces by a change of metric and their imbedding in hilbert space. *Annals of Mathematics*, 38(4):787–793, 1937.

L. G. Shapiro and R.M. Haralick. A metric for comparing relational descriptions. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 7:90–94, 1985.

D. Shasha and K. Zhang. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18:245–262, 1989.

N.Z. Shor. *Minimization Methods for non-differentiable Functions*. Springer, 1985.

R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Annals of Mathematical Statistics*, 35(2):876 – 879, 1964.

B. Taskar. *Learning Structured Prediction Models: A Large Margin Approach*. PhD thesis, Stanford University, 2004.

S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Elsevier, 4 edition, 2009.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *In Proceedings of the 21st International Conference on Machine Learning (ICML'04)*, 2004.

U. von Luxburg and O. Bousquet. Distance-based classification with Lipschitz functions. *Journal of Machine Learning Research*, 5:669–665, 2004.

C. Watson and C. Wilson. NIST Special Database 4, Fingerprint Database. National Institute of Standards and Technology, 1992.

W. Xie and N.V. Sahinidis. A reduction-based exact algorithm for the contact map overlap problem. *Journal of Computational Biology*, 14(5):637–654, 2007.

X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Second IEEE International Conference on Data Mining (ICDM'02)*, pages 721–724, 2002.

K. Zhang. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15:205–222, 1996.

Journal of Machine Learning Research 10 (2009) 2715-2740 Submitted 5/09; Revised 10/09; Published 12/09

Learning Halfspaces with Malicious Noise

Adam R. Klivans KLIVANS@CS.UTEXAS.EDU
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712 USA

Philip M. Long PLONG@GOOGLE.COM
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043 USA

Rocco A. Servedio ROCCO@CS.COLUMBIA.EDU
Department of Computer Science
Columbia University
1214 Amsterdam Avenue, Mailcode 0401
New York, NY 10027, USA

Editor: Manfred K. Warmuth

Abstract

We give new algorithms for learning halfspaces in the challenging *malicious noise* model, where an adversary may corrupt both the labels and the underlying distribution of examples. Our algorithms can tolerate malicious noise rates exponentially larger than previous work in terms of the dependence on the dimension n, and succeed for the fairly broad class of all isotropic log-concave distributions.

We give $\mathrm{poly}(n, 1/\varepsilon)$-time algorithms for solving the following problems to accuracy ε:

- Learning origin-centered halfspaces in $\mathbf{R}^n$ with respect to the uniform distribution on the unit ball with malicious noise rate $\eta = \Omega(\varepsilon^2/\log(n/\varepsilon))$. (The best previous result was $\Omega(\varepsilon/(n\log(n/\varepsilon))^{1/4})$.)

- Learning origin-centered halfspaces with respect to any isotropic log-concave distribution on $\mathbf{R}^n$ with malicious noise rate $\eta = \Omega(\varepsilon^3/\log^2(n/\varepsilon))$. This is the first efficient algorithm for learning under isotropic log-concave distributions in the presence of malicious noise.

We also give a $\mathrm{poly}(n, 1/\varepsilon)$-time algorithm for learning origin-centered halfspaces under any isotropic log-concave distribution on $\mathbf{R}^n$ in the presence of *adversarial label noise* at rate $\eta = \Omega(\varepsilon^3/\log(1/\varepsilon))$. In the adversarial label noise setting (or agnostic model), labels can be noisy, but not example points themselves. Previous results could handle $\eta = \Omega(\varepsilon)$ but had running time exponential in an unspecified function of $1/\varepsilon$.

Our analysis crucially exploits both concentration and anti-concentration properties of isotropic log-concave distributions. Our algorithms combine an iterative outlier removal procedure using Principal Component Analysis together with "smooth" boosting.

Keywords: PAC learning, noise tolerance, malicious noise, agnostic learning, label noise, halfspace learning, linear classifiers

1. Introduction

A *halfspace* is a Boolean-valued function of the form $f = \mathrm{sign}(\sum_{i=1}^{n} w_i x_i - \theta)$. Learning halfspaces in the presence of noisy data is a fundamental problem in machine learning. In addition to its practical relevance, the problem has connections to many well-studied topics such as kernel methods (Shawe-Taylor and Cristianini, 2000), cryptographic hardness of learning (Klivans and Sherstov, 2006), hardness of approximation (Feldman et al., 2006; Guruswami and Raghavendra, 2006), learning Boolean circuits (Blum et al., 1997), and additive/multiplicative update learning algorithms (Littlestone, 1991; Freund and Schapire, 1999).

Learning an unknown halfspace from correctly labeled (non-noisy) examples is one of the best-understood problems in learning theory, with work dating back to the famous Perceptron algorithm of the 1950s (Rosenblatt, 1958) and a range of efficient algorithms known for different settings (Novikoff, 1962; Littlestone, 1987; Blumer et al., 1989; Maass and Turan, 1994). Much less is known, however, about the more difficult problem of learning halfspaces in the presence of noise.

Important progress was made by Blum et al. (1997) who gave a polynomial-time algorithm for learning a halfspace under *classification noise*. In this model each label is flipped independently with some fixed probability; the noise does not affect the actual example points themselves, which are generated according to an arbitrary probability distribution over $\mathbf{R}^n$.

In the current paper we consider a much more challenging *malicious noise* model. In this model, introduced by Valiant (1985) (see also Kearns and Li 1993), there is an unknown target function f and distribution $\mathcal{D}$ over examples. Each time the learner receives an example, independently with probability $1-\eta$ it is drawn from $\mathcal{D}$ and labeled correctly according to f, but with probability η it is an arbitrary pair (x,y) which may be generated by an omniscient adversary. The parameter η is known as the "noise rate."

Malicious noise is a notoriously difficult model with few positive results. It was already shown by Kearns and Li (1993) that for essentially all concept classes, it is information-theoretically impossible to learn to accuracy $1-\varepsilon$ if the noise rate η is greater than $\varepsilon/(1+\varepsilon)$. Indeed, known algorithms for learning halfspaces (Servedio, 2003; Kalai et al., 2008) or even simpler target functions (Mansour and Parnas, 1998) with malicious noise typically make strong assumptions about the underlying distribution $\mathcal{D}$, and can learn to accuracy $1-\varepsilon$ only for noise rates η much smaller than ε. We describe the most closely related work that we know of in Section 1.2.

In this paper we consider learning under the uniform distribution on the unit ball in $\mathbf{R}^n$, and more generally under any isotropic log-concave distribution. The latter is a fairly broad class of distributions that includes spherical Gaussians and uniform distributions over a wide range of convex sets. Our algorithms can learn from malicious noise rates that are quite high, as we now describe.

1.1 Main Results

Our first result is an algorithm for learning halfspaces in the malicious noise model with respect to the uniform distribution on the n-dimensional unit ball:

Theorem 1 *There is a* $\mathrm{poly}(n, 1/\varepsilon)$*-time algorithm that learns origin-centered halfspaces to accuracy* $1-\varepsilon$ *with respect to the uniform distribution on the unit ball in n dimensions in the presence of malicious noise at rate* $\eta = \Omega(\varepsilon^2/\log(n/\varepsilon))$.

The condition on η is expressed using Ω and not O because we are showing that a weak upper bound on the noise rate suffices to achieve accuracy $1-\varepsilon$.

Via a more sophisticated algorithm, we can learn in the presence of malicious noise under any isotropic log-concave distribution:

Theorem 2 *There is a* $\mathrm{poly}(n, 1/\varepsilon)$*-time algorithm that learns origin-centered halfspaces to accuracy* $1-\varepsilon$ *with respect to any isotropic log-concave distribution over* $\mathbf{R}^n$ *and can tolerate malicious noise at rate* $\eta = \Omega(\varepsilon^3/\log^2(n/\varepsilon))$.

We are not aware of any previous polynomial-time algorithms for learning under isotropic log-concave distributions in the presence of malicious noise.

Finally, we also consider a related noise model known as *adversarial label noise*. In this model there is a fixed probability distribution P over $\mathbf{R}^n \times \{-1, 1\}$ (i.e., over labeled examples) for which a $1-\eta$ fraction of draws are labeled according to an unknown halfspace. The marginal distribution over $\mathbf{R}^n$ is assumed to be isotropic log-concave; so the idea is that an "adversary" chooses an η fraction of examples to mislabel, but unlike the malicious noise model she cannot change the (isotropic log-concave) distribution of the actual example points in $\mathbf{R}^n$. Learning with adversarial label noise is clearly harder than with independent misclassification noise—the ability to choose which labels to corrupt allows the adversary to coordinate their effects to an extent.

For the adversarial label noise model we prove:

Theorem 3 *There is a* $\mathrm{poly}(n, 1/\varepsilon)$*-time algorithm that learns origin-centered halfspaces to accuracy* $1-\varepsilon$ *with respect to any isotropic log-concave distribution over* $\mathbf{R}^n$ *and can tolerate adversarial label noise at rate* $\eta = \Omega(\varepsilon^3/\log(1/\varepsilon))$.

1.2 Previous Work

Our work builds on a number of lines of research.

1.2.1 MALICIOUS NOISE

General-purpose tools developed by Kearns and Li (1993) (see also Kearns et al. 1994) directly imply that halfspaces can be learned for any distribution over the domain in randomized $\mathrm{poly}(n, 1/\varepsilon)$ time with malicious noise at a rate $\Omega(\varepsilon/n)$; the algorithm repeatedly picks a random subsample of the training data, hoping to miss all the noisy examples. Kannan (see Arora et al. 1993) devised a deterministic algorithm with a $\Omega(\varepsilon/n)$ bound that repeatedly exploits Helly's Theorem to find a group of $n+1$ examples that includes a noisy example, then removes the group. Kalai et al. (2008) showed that the $\mathrm{poly}(n, 1/\varepsilon)$-time averaging algorithm (Servedio, 2001) tolerates noise at a rate $\Omega(\varepsilon/\sqrt{n})$ when the distribution is uniform. They also described an improvement to $\tilde{\Omega}(\varepsilon/n^{1/4})$ based on the observation that uniform examples will tend to be well-separated, so that pairs of examples that are too close to one another can be removed.

1.2.2 ADVERSARIAL LABEL NOISE

Kalai, et al. showed that if the distribution over the instances is uniform over the unit ball, the averaging algorithm tolerates adversarial label noise at a rate $\Omega(\varepsilon/\sqrt{\log(1/\varepsilon)})$ in $\mathrm{poly}(n, 1/\varepsilon)$ time. (In that paper, learning in the presence of adversarial label noise was called "agnostic learning".) They also described an algorithm that fits low-degree polynomials that tolerates noise at a rate within an additive ε of the accuracy, but in $\mathrm{poly}\left(n^{1/\varepsilon^4}\right)$ time; for log-concave distributions, their algorithm

took poly$\left(n^{d(1/\varepsilon)}\right)$ time, for an unspecified function d. The latter algorithm does not require that the distribution is isotropic, as ours does.

1.2.3 ROBUST PCA

Independently of this work, Xu et al. (2009) designed and analyzed an algorithm that performs principal component analysis when some of the examples are corrupted arbitrarily, as in the malicious noise model studied here. Also, the thesis of Brubaker (2009) presents a "Robust PCA" algorithm which is a PCA variant aimed at ameliorating the effects of noisy examples.

1.3 Techniques

Here is a high-level description of the main techniques in our analysis.

1.3.1 OUTLIER REMOVAL

Consider first the simplest problem of learning an origin-centered halfspace with respect to the uniform distribution on the n-dimensional ball. A natural idea is to use a simple "averaging" algorithm that takes the vector average of the positive examples it receives and uses this as the normal vector of its hypothesis halfspace. Servedio (2001) analyzed this algorithm for the random classification noise model, and Kalai et al. (2008) extended the analysis to the adversarial label noise model.

Intuitively the "averaging" algorithm can only tolerate low malicious noise rates because the adversary can generate noisy examples which "pull" the average vector far from its true location. Our main insight is that the adversary does this most effectively when the noisy examples are coordinated to pull in roughly the same direction. We use a form of outlier detection based on Principal Component Analysis to detect such coordination. This is done by computing the direction $\mathbf{w}$ of maximal variance of the data set; if the variance in direction $\mathbf{w}$ is suspiciously large, we remove from the sample all points $\mathbf{x}$ for which $(\mathbf{w} \cdot \mathbf{x})^2$ is large. Our analysis shows that this causes many noisy examples, and only a few non-noisy examples, to be removed.

We repeat this process until the variance in every direction is not too large. (This cannot take too many stages since many noisy examples are removed in each stage.) While some noisy examples may remain, we show that their scattered effects cannot hurt the algorithm much.

Thus, in a nutshell, our overall algorithm for the uniform distribution is to first do outlier removal[1] by an iterated PCA-type procedure, and then simply run the averaging algorithm on the remaining "cleaned-up" data set.

1.3.2 EXTENDING TO LOG-CONCAVE DISTRIBUTIONS VIA SMOOTH BOOSTING

We are able to show that the iterative outlier removal procedure described above is useful for isotropic log-concave distributions as well as the uniform distribution: if examples are removed in a given stage, then many of the removed examples are noisy and only a few are non-noisy (the analysis here uses concentration bounds for isotropic log-concave distributions). However, even if there were no noise in the data, the average of the positive examples under an isotropic log-concave

1. We note briefly that the sophisticated outlier removal techniques of Blum et al. (1997) and Dunagan and Vempala (2004) do not seem to be useful in our setting; those works deal with a strong notion of outliers, which is such that no point on the unit ball can be an outlier if a significant fraction of points are uniformly distributed on the unit ball.

distribution need not give a high-accuracy hypothesis. Thus the averaging algorithm alone will not suffice after outlier removal.

To get around this, we show that after outlier removal the average of the positive examples gives a (real-valued) *weak* hypothesis that has some nontrivial predictive accuracy. (Interestingly, the proof of this relies heavily on *anti*-concentration properties of isotropic log-concave distributions!) A natural approach is then to use a boosting algorithm to convert this weak learner into a strong learner. This is not entirely straightforward because boosting "skews" the distribution of examples; this has the undesirable effects of both increasing the effective malicious noise rate, and causing the distribution to no longer be isotropic log-concave. However, by using a "smooth" boosting algorithm (Servedio, 2003) that skews the distribution as little as possible, we are able to control these undesirable effects and make the analysis go through. (The extra factor of ε in the bound of Theorem 2 compared with Theorem 1 comes from the fact that the boosting algorithm constructs "$1/\varepsilon$-skewed" distributions.)

We note that our approach of using smooth boosting is reminiscent of earlier work (Servedio, 2002, 2003), but the current algorithm goes well beyond that. Servedio (2002) did not consider a noisy scenario, and Servedio (2003) only considered the averaging algorithm without any outlier removal as the weak learner (and thus could only handle quite low rates of malicious noise in our isotropic log-concave setting).

1.3.3 TOLERATING ADVERSARIAL LABEL NOISE

Finally, our results for learning under isotropic log-concave distributions with adversarial label noise are obtained using a similar approach. The algorithm here is in fact simpler than the malicious noise algorithm: since the adversarial label noise model does not allow the adversary to alter the distribution of the examples in $\mathbf{R}^n$, we can dispense with the outlier removal and simply use smooth boosting with the averaging algorithm as the weak learner. (This is why we get a slightly better quantitative bound in Theorem 3 than Theorem 2).

1.3.4 ORGANIZATION

For completeness we review the precise definitions of isotropic log-concave distributions and the various learning models in Section 2. We present the simpler and more easily understood uniform distribution analysis in Section 3. We extend the algorithm and analysis to isotropic log-concave distributions in Section 4. Learning with adversarial label noise is treated in Section 5. We conclude in Section 6.

2. Definitions and Preliminaries

In this section, we provide some definitions and lemmas that will be used throughout the paper.

2.1 Learning with Malicious Noise

Given a probability distribution $\mathcal{D}$ over $\mathbf{R}^n$, and a target function $f : \mathbf{R}^n \to \{-1, 1\}$, we define the oracle $\mathrm{EX}_\eta(f, \mathcal{D})$ as follows:

- with probability $1 - \eta$ the oracle draws $\mathbf{x}$ according to $\mathcal{D}$, and outputs $(\mathbf{x}, f(\mathbf{x}))$, and

- with probability η the oracle outputs an arbitrary $(\mathbf{x}, y)$ pair. This "noisy" example can be thought of as being generated adversarially and can depend on the state of the learning algorithm and previous draws from the oracle.

Given a data set drawn from $\mathrm{EX}_\eta(f, \mathcal{D})$, we often refer to the examples $(\mathbf{x}, f(\mathbf{x}))$ (that came from $\mathcal{D}$) as "clean" examples and the remaining examples $(\mathbf{x}, y)$ as "dirty" examples.

For a set $\mathcal{S}$ of probability distributions and a set F of possible target functions, we say that a learning algorithm A learns F to accuracy $1-\varepsilon$ with respect to $\mathcal{S}$ in the presence of malicious noise at a rate η if the following holds: for any $f \in F$, and $\mathcal{D} \in \mathcal{S}$, given access to $\mathrm{EX}_\eta(f, \mathcal{D})$, with probability at least $1/2$, the output hypothesis h generated by A satisfies $\Pr_{\mathbf{x}\sim\mathcal{D}}[h(\mathbf{x}) \neq f(\mathbf{x})] \leq \varepsilon$. (The probability of success may be amplified arbitrarily close to 1 using standard techniques (Haussler et al., 1991).)

Since scaling $\mathbf{x}$ by a positive constant does not affect its classification by a linear classifier, drawing examples uniformly from the unit ball is equivalent to drawing them uniformly from the surface $\mathbb{S}^{n-1}$ of the unit sphere. When this is the distribution, we may also assume w.l.o.g. that even noisy examples $(\mathbf{x}, y)$ have $\mathbf{x} \in \mathbb{S}^{n-1}$—this is simply because a learning algorithm can trivially identify and ignore any noisy example $(\mathbf{x}, y)$ that has $\|\mathbf{x}\| \neq 1$.

2.2 Log-concave Distributions

A probability distribution over $\mathbf{R}^n$ is said to be *log-concave* if its density function is $\exp(-\psi(\mathbf{x}))$ for a convex function ψ.

A probability distribution over $\mathbf{R}^n$ is *isotropic* if the mean of the distribution is 0 and the covariance matrix is the identity, that is, $\mathbf{E}[x_i x_j] = 1$ for $i = j$ and 0 otherwise.

Isotropic log-concave (henceforth abbreviated i.l.c.) distributions are a fairly broad class of distributions. It is well known that any distribution induced by taking a uniform distribution over an arbitrary convex set and applying a suitable linear transformation to make it isotropic is then isotropic and log-concave. For an excellent treatment on basic properties of log-concave distributions, see Lovász and Vempala (2007).

We will use the following facts:

Lemma 4 (Lovász and Vempala 2007) *Let $\mathcal{D}$ be an isotropic log-concave distribution over $\mathbf{R}^n$ and $\mathbf{a} \in \mathbb{S}^{n-1}$ any direction. Then for $\mathbf{x}$ drawn according to $\mathcal{D}$, the distribution of $\mathbf{a}\cdot\mathbf{x}$ is an isotropic log-concave distribution over $\mathbf{R}$.*

Lemma 5 (Lovász and Vempala 2007) *Any isotropic log-concave distribution $\mathcal{D}$ over $\mathbf{R}^n$ has light tails,*

$$\Pr_{\mathbf{x}\sim\mathcal{D}}[||\mathbf{x}|| > \beta\sqrt{n}] \leq e^{-\beta+1}.$$

If $n = 1$, the density of $\mathcal{D}$ is bounded:

$$\Pr_{x\sim\mathcal{D}}[\mathbf{x} \in [a,b]] \leq |b-a|.$$

3. The Uniform Distribution and Malicious Noise

In this section we prove Theorem 1. As described above, our algorithm first does outlier removal using PCA and then applies the "averaging algorithm."

We may assume throughout that the noise rate η is smaller than some absolute constant, and that the dimension n is larger than some absolute constant.

3.1 The Algorithm: Removing Outliers and Averaging

Consider the following Algorithm A_{mu}:

Algorithm A_{mu}:

1. Draw a sample S of $m = \mathrm{poly}(n/\varepsilon)$ many examples from the malicious oracle.
2. Identify the direction $\mathbf{w} \in \mathbb{S}^{n-1}$ that maximizes

$$\sigma_{\mathbf{w}}^2 \stackrel{def}{=} \sum_{(\mathbf{x},y)\in S} (\mathbf{w}\cdot\mathbf{x})^2.$$

   If $\sigma_{\mathbf{w}}^2 < \frac{10m\log m}{n}$ then go to Step 4 otherwise go to Step 3.
3. Remove from S every example that has $(\mathbf{w}\cdot\mathbf{x})^2 \geq \frac{10\log m}{n}$. Go to Step 2.
4. For the examples S that remain let $\mathbf{v} = \frac{1}{|S|}\sum_{(\mathbf{x},y)\in S} y\mathbf{x}$ and output the linear classifier $h_{\mathbf{v}}$ defined by $h_{\mathbf{v}}(\mathbf{x}) = \mathrm{sgn}(\mathbf{v}\cdot\mathbf{x})$.

We first observe that Step 2 can be carried out in polynomial time:

Lemma 6 *There is a polynomial-time algorithm that, given a finite collection S of points in $\mathbf{R}^n$, outputs $\mathbf{w} \in \mathbb{S}^{n-1}$ that maximizes $\sum_{\mathbf{x}\in S}(\mathbf{w}\cdot\mathbf{x})^2$.*

Proof. By applying Lagrange multipliers, we can see that the optimal $\mathbf{w}$ is an eigenvector of $A = \sum_{\mathbf{x}\in S}\mathbf{x}\mathbf{x}^T$. Further, if λ is the eigenvalue of $\mathbf{w}$, then $\sum_{\mathbf{x}\in S}(\mathbf{w}\cdot\mathbf{x})^2 = \mathbf{w}^T A\mathbf{w} = \mathbf{w}^T(\lambda\mathbf{w}) = \lambda$. The eigenvector $\mathbf{w}$ with the largest eigenvalue can be found in polynomial time (see, e.g., Jolliffe 2002). ∎

Before embarking on the analysis we establish a terminological convention. Much of our analysis deals with high-probability statements over the draw of the m-element sample S; it is straightforward but quite cumbersome to explicitly keep track of all of the failure probabilities. Thus we write "with high probability" (or "w.h.p.") in various places below as a shorthand for "with probability at least $1 - 1/\mathrm{poly}(n/\varepsilon)$." The interested reader can easily verify that an appropriate $\mathrm{poly}(n/\varepsilon)$ choice of m makes all the failure probabilities small enough so that the entire algorithm succeeds with probability at least $1/2$ as required.

3.2 Properties of the Clean Examples

In this subsection we establish properties of the clean examples that were sampled in Step 1 of A_{mu}. The first says that no direction has much more variance than the expected variance of $1/n$:

Lemma 7 *W.h.p. over a random draw of ℓ clean examples S_{clean}, we have*

$$\max_{a\in\mathbb{S}^{n-1}}\left\{\frac{1}{\ell}\sum_{(\mathbf{x},y)\in S_{\mathrm{clean}}}(\mathbf{a}\cdot\mathbf{x})^2\right\} \leq \frac{1}{n} + \sqrt{\frac{O(n+\log\ell)}{\ell}}.$$

Proof. The proof uses standard tools from VC theory and is in Appendix A. ∎

The next lemma says that in fact no direction has too many clean examples lying far out in that direction:

Lemma 8 *For any $\beta > 0$ and $\kappa > 1$, if S_{clean} is a random set of $\ell \geq \frac{O(1)\cdot n^2\beta^2 e^{\beta^2 n/2}}{(1+\kappa)\ln(1+\kappa)}$ clean examples then w.h.p. we have*

$$\max_{\mathbf{a}\in\mathbb{S}^{n-1}} \frac{1}{\ell}|\{\mathbf{x} \in S_{\text{clean}} : (\mathbf{a}\cdot\mathbf{x})^2 > \beta^2\}| \;\leq\; (1+\kappa)e^{-\beta^2 n/2}.$$

Proof. In Appendix B. ∎

3.3 What is Removed

In this section, we provide bounds on the number of clean and dirty examples removed in Step 3.

The first bound is a Corollary of Lemma 8.

Corollary 9 *W.h.p. over the random draw of the m-element sample S, the number of clean examples removed during any one execution of Step 3 in A_{mu} is at most $6n\log m$.*

Proof. Since the noise rate η is sufficiently small, w.h.p. the number ℓ of clean examples is at least (say) $m/2$. We would like to apply Lemma 8 with $\kappa = 5\ell^4 n\log\ell$ and $\beta = \sqrt{\frac{10\log m}{n}}$, and indeed we may do this because we have

$$\frac{O(1)\cdot n^2\beta^2 e^{\beta^2 n/2}}{(1+\kappa)\ln(1+\kappa)} \leq \frac{O(1)\cdot n(\log m)m^5}{(1+\kappa)\ln(1+\kappa)} \leq O\left(\frac{m}{\log m}\right) \leq \frac{m}{2} \leq \ell$$

for n sufficiently large. Since clean points are only removed if they have $(\mathbf{a}\cdot\mathbf{x})^2 > \beta^2$, Lemma 8 gives us that the number of clean points removed is at most

$$m(1+\kappa)e^{-\beta^2 n/2} \leq 6m^5 n\log(\ell)/m^5 \leq 6n\log m.$$

∎

The counterpart to Corollary 9 is the following lemma. It tells us that if examples are removed in Step 3, then there must be many *dirty* examples removed. It exploits the fact that Lemma 7 bounds the variance in *all* directions $\mathbf{a}$, so that it can be reused to reason about what happens in different executions of step 3.

Lemma 10 *W.h.p. over the random draw of S, whenever A_{mu} executes step 3, it removes at least $\frac{4m\log m}{n}$ noisy examples from S_{dirty}, the set of dirty examples in S.*

Proof. As stated earlier we may assume that $\eta \leq 1/4$. This implies that w.h.p. the fraction $\widehat{\eta}$ of noisy examples in the initial set S is at most $1/2$. Finally, Lemma 7 implies that $m = \tilde{\Omega}(n^3)$ suffices for it to be the case that w.h.p., for all $\mathbf{a} \in \mathbb{S}^{n-1}$, for the original multiset S_{clean} of clean examples drawn in step 1, we have

$$\sum_{(\mathbf{x},y)\in S_{\text{clean}}} (\mathbf{a}\cdot\mathbf{x})^2 \leq \frac{2m}{n}. \tag{1}$$

We shall say that a random sample S that satisfies all these requirements is "reasonable". We will show that for any reasonable data set, the number of noisy examples removed during the execution of step 3 of A_{mu} is at least $\frac{4m\log m}{n}$.

If we remove examples using direction $\mathbf{w}$ then it means $\sum_{(\mathbf{x},y)\in S}(\mathbf{w}\cdot\mathbf{x})^2 \geq \frac{10m\log m}{n}$. Since S is reasonable, by (1) the contribution to the sum from the clean examples that survived to the current stage is at most $2m/n$ so we must have

$$\sum_{(\mathbf{x},y)\in S_{\text{dirty}}}(\mathbf{w}\cdot\mathbf{x})^2 \geq 10m\log(m)/n - 2m/n > 9m\log(m)/n.$$

Let us decompose S_{dirty} into $N\cup F$ where N ("near") consists of those points x s.t. $(\mathbf{w}\cdot\mathbf{x})^2 \leq 10\log(m)/n$ and F ("far") is the remaining points for which $(\mathbf{w}\cdot\mathbf{x})^2 > 10\log(m)/n$. Since $|N| \leq |S_{\text{dirty}}| \leq \widehat{\eta}m$, (any dirty examples removed in earlier rounds will only reduce the size of S_{dirty}) we have

$$\sum_{(\mathbf{x},y)\in N}(\mathbf{w}\cdot\mathbf{x})^2 \leq (\widehat{\eta}m)10\log(m)/n$$

and so

$$|F| \geq \sum_{(\mathbf{x},y)\in F}(\mathbf{w}\cdot\mathbf{x})^2 \geq 9m\log(m)/n - (\widehat{\eta}m)10\log(m)/n \geq 4m\log(m)/n$$

(the last line used the fact that $\widehat{\eta} < 1/2$). Since the points in F are removed in Step 3, the lemma is proved. ∎

3.4 Exploiting Limited Variance in Any Direction

In this section, we show that if all directional variances are small, then the algorithm's final hypothesis will have high accuracy.

We first recall a simple lemma which shows that a sample of "clean" examples results in a high-accuracy hypothesis for the averaging algorithm:

Lemma 11 (Servedio 2001) *Suppose $\mathbf{x}_1,\ldots,\mathbf{x}_m$ are chosen uniformly at random from $\mathbb{S}^{n-1}$, and a target weight vector $\mathbf{u}\in\mathbb{S}^{n-1}$ produces labels $y_1 = \text{sign}(\mathbf{u}\cdot\mathbf{x}_1),\ldots,y_m = \text{sign}(\mathbf{u}\cdot\mathbf{x}_m)$. Let $\mathbf{v} = \frac{1}{m}\sum_{t=1}^{m} y_t\mathbf{x}_t$. Then w.h.p. $\mathbf{u}\cdot\mathbf{v} = \Omega(\frac{1}{\sqrt{n}})$, while $||\mathbf{v} - (\mathbf{u}\cdot\mathbf{v})\mathbf{u}|| = O(\sqrt{\log(n)/m})$.*

Now we can state Lemma 12.

Lemma 12 *Let $S = S_{\text{clean}}\cup S_{\text{dirty}}$ be the sample of m examples drawn from the noisy oracle $\text{EX}_\eta(f,\mathcal{U})$. Let*

- *S'_{clean} be those clean examples that were never removed during step 3 of A_{mu},*
- *S'_{dirty} be those dirty examples that were never removed during step 3 of A_{mu},*
- *$\eta' = \frac{|S'_{\text{dirty}}|}{|S'_{\text{clean}}\cup S'_{\text{dirty}}|}$, that is, the fraction of dirty examples among the examples that survive step 3, and*
- *$\alpha = \frac{|S_{\text{clean}} - S'_{\text{clean}}|}{|S'_{\text{clean}}\cup S'_{\text{dirty}}|}$, the ratio of the number of clean points that were erroneously removed to the size of the final surviving data set.*

Let $S' \stackrel{def}{=} S'_{\text{clean}} \cup S'_{\text{dirty}}$. Suppose that $|S'| \geq m/2$ (i.e., fewer than half the total points were removed) and that, for every direction $\mathbf{w} \in \mathbb{S}^{n-1}$ we have

$$\sigma^2_{\mathbf{w}} \stackrel{def}{=} \sum_{(\mathbf{x},y)\in S'} (\mathbf{w}\cdot\mathbf{x})^2 \leq \frac{10m\log m}{n}.$$

Then w.h.p. over the draw of S, the halfspace with normal vector $\mathbf{v} \stackrel{def}{=} \frac{1}{|S'|}\sum_{(\mathbf{x},y)\in S'} y\mathbf{x}$ has error rate

$$O\left(\sqrt{\eta' \log m} + \alpha\sqrt{n} + \sqrt{\frac{n\log n}{m}}\right).$$

Proof. The claimed bound is trivial unless $\eta' \leq o(1)/\log m$ and $\alpha \leq o(1)/\sqrt{n}$, so we shall freely use these bounds in what follows.

Let $\mathbf{u}$ be the unit length normal vector for the target halfspace. Let $\mathbf{v}_{\text{clean}}$ be the average of *all* the clean examples, $\mathbf{v}'_{\text{dirty}}$ be the average of the dirty (noisy) examples that were not deleted (i.e., the examples in S'_{dirty}), and $\mathbf{v}_{\text{del}}$ be the average of the clean examples that were deleted. Then

$$\begin{aligned}
\mathbf{v} &= \frac{1}{|S'_{\text{clean}} \cup S'_{\text{dirty}}|} \sum_{(\mathbf{x},y)\in S'_{\text{clean}}\cup S'_{\text{dirty}}} y\mathbf{x} \\
&= \frac{1}{|S'_{\text{clean}} \cup S'_{\text{dirty}}|} \left(\left(\sum_{(\mathbf{x},y)\in S_{\text{clean}}} y\mathbf{x}\right) + \left(\sum_{(\mathbf{x},y)\in S'_{\text{dirty}}} y\mathbf{x}\right) - \left(\sum_{(\mathbf{x},y)\in S_{\text{clean}}-S'_{\text{clean}}} y\mathbf{x}\right)\right) \\
\mathbf{v} &= (1-\eta'+\alpha)\mathbf{v}_{\text{clean}} + \eta'\mathbf{v}'_{\text{dirty}} - \alpha\mathbf{v}_{\text{del}}.
\end{aligned}$$

Let us begin by exploiting the bound on the variance in every direction to bound the length of $\mathbf{v}'_{\text{dirty}}$. For any $\mathbf{w} \in \mathbb{S}^{n-1}$ we know that

$$\sum_{(\mathbf{x},y)\in S'} (\mathbf{w}\cdot\mathbf{x})^2 \leq \frac{10m\log m}{n}, \quad \text{and hence} \quad \sum_{(\mathbf{x},y)\in S'_{\text{dirty}}} (\mathbf{w}\cdot\mathbf{x})^2 \leq \frac{10m\log m}{n}$$

since $S'_{\text{dirty}} \subseteq S'$. Since $|S'_{\text{dirty}}| \leq \eta' m$, the fact that $||\mathbf{r}||_1 \leq \sqrt{k}||\mathbf{r}||_2$ for any vector $\mathbf{r} \in \mathbf{R}^k$ gives

$$\sum_{(\mathbf{x},y)\in S'_{\text{dirty}}} |\mathbf{w}\cdot\mathbf{x}| \leq \sqrt{\frac{10m|S'_{\text{dirty}}|\log m}{n}}.$$

Taking $\mathbf{w}$ to be the unit vector in the direction of $\mathbf{v}'_{\text{dirty}}$, we have $\|\mathbf{v}'_{\text{dirty}}\| =$

$$\mathbf{w}\cdot\mathbf{v}'_{\text{dirty}} = \mathbf{w}\cdot\frac{1}{|S'_{\text{dirty}}|}\sum_{(\mathbf{x},y)\in S'_{\text{dirty}}} y\mathbf{x} \leq \frac{1}{|S'_{\text{dirty}}|}\sum_{(\mathbf{x},y)\in S'_{\text{dirty}}} |\mathbf{w}\cdot\mathbf{x}| \leq \sqrt{\frac{10m\log m}{|S'_{\text{dirty}}|n}}. \tag{2}$$

Because the domain distribution is uniform, the error of $h_{\mathbf{v}}$ is proportional to the angle between $\mathbf{v}$ and $\mathbf{u}$, in particular,

$$\Pr[h_{\mathbf{v}} \neq f] = \frac{1}{\pi}\arctan\left(\frac{||\mathbf{v}-(\mathbf{v}\cdot\mathbf{u})\mathbf{u}||}{\mathbf{u}\cdot\mathbf{v}}\right) \leq (1/\pi)\frac{||\mathbf{v}-(\mathbf{v}\cdot\mathbf{u})\mathbf{u}||}{\mathbf{u}\cdot\mathbf{v}}. \tag{3}$$

We have that $||\mathbf{v}-(\mathbf{v}\cdot\mathbf{u})\mathbf{u}||$ equals

$$||(1-\eta'+\alpha)(\mathbf{v}_{\text{clean}}-(\mathbf{v}_{\text{clean}}\cdot\mathbf{u})\mathbf{u})+\eta'(\mathbf{v}'_{\text{dirty}}-(\mathbf{v}'_{\text{dirty}}\cdot\mathbf{u})\mathbf{u})-\alpha(\mathbf{v}_{\text{del}}-(\mathbf{v}_{\text{del}}\cdot\mathbf{u})\mathbf{u})||$$
$$\leq 2||\mathbf{v}_{\text{clean}}-(\mathbf{v}_{\text{clean}}\cdot\mathbf{u})\mathbf{u}||+\eta'||\mathbf{v}'_{\text{dirty}}||+\alpha||\mathbf{v}_{\text{del}}||$$

where we have used the triangle inequality and the fact that α,η are "small." Lemma 11 lets us bound the first term in the sum by $O(\sqrt{\log(n)/m})$, and the fact that $\mathbf{v}_{\text{del}}$ is an average of vectors of length 1 lets us bound the third by α. For the second term, Equation (2) gives us

$$\eta'\|\mathbf{v}'_{\text{dirty}}\|\leq\sqrt{\frac{10m(\eta')^2\log m}{|S'_{\text{dirty}}|n}}=\sqrt{\frac{10m\eta'\log m}{|S'|n}}\leq\sqrt{\frac{20\eta'\log m}{n}},$$

where for the last equality we used $|S'|\geq m/2$. We thus get

$$||\mathbf{v}-(\mathbf{v}\cdot\mathbf{u})\mathbf{u}||\leq O\left(\sqrt{\log(n)/m}\right)+\sqrt{20\eta'\log(m)/n}+\alpha. \tag{4}$$

Now we consider the denominator of (3). We have

$$\mathbf{u}\cdot\mathbf{v}=(1-\eta'+\alpha)(\mathbf{u}\cdot\mathbf{v}_{\text{clean}})+\eta'\mathbf{u}\cdot\mathbf{v}'_{\text{dirty}}-\alpha\mathbf{u}\cdot\mathbf{v}_{\text{del}}.$$

Similar to the above analysis, we again use Lemma 11 (but now the lower bound $\mathbf{u}\cdot\mathbf{v}\geq\Omega(1/\sqrt{n})$), Equation (2), and the fact that $||\mathbf{v}_{\text{del}}||\leq 1$. Since α and η' are "small," we get that there is an absolute constant c such that $\mathbf{u}\cdot\mathbf{v}\geq c/\sqrt{n}-\sqrt{20\eta'\log(m)/n}-\alpha$. Combining this with (4) and (3), we get

$$\Pr[h_{\mathbf{v}}\neq f] \leq \frac{O\left(\sqrt{\frac{\log n}{m}}\right)+\sqrt{\frac{20\eta'\log m}{n}}+\alpha}{\pi\left(\frac{c}{\sqrt{n}}-\sqrt{\frac{20\eta'\log m}{n}}-\alpha\right)}=O\left(\sqrt{\frac{n\log n}{m}}+\sqrt{\eta'\log m}+\alpha\sqrt{n}\right).$$

■

3.5 Proof of Theorem 1

By Corollary 9, w.h.p. each outlier removal stage removes at most $6n\log m$ clean points.

Since, by Lemma 10, each outlier removal stage removes at least $\frac{4m\log m}{n}$ noisy examples, there must be at most $O(n/(\log m))$ such stages. Consequently the total number of clean examples removed across all stages is $O(n^2)$. Since w.h.p. the initial number of clean examples is at least $3m/4$, this means that the final data set (on which the averaging algorithm is run) contains at least $3m/4-O(n^2)$ clean examples, and hence at least $3m/4-O(n^2)$ examples in total. The condition $m\gg n^2$ means that the number of surviving examples will be at least $m/2$. Consequently the value of α from Lemma 12 after the final outlier removal stage (the ratio of the total number of clean examples deleted, to the total number of surviving examples) is at most $\frac{O(n^2)}{m}$.

The standard Hoeffding bound implies that w.h.p. the actual fraction of noisy examples in the original sample S is at most $\eta+\sqrt{O(\log m)/m}$. It is easy to see that w.h.p. the fraction of dirty examples does not increase (since each stage of outlier removal removes more dirty points than clean points, for a suitably large $\text{poly}(n/\varepsilon)$ value of m), and thus the fraction η' of dirty examples among the remaining examples after the final outlier removal stage is at most $\eta+\sqrt{O(\log m)/m}$.

Applying Lemma 12, for a suitably large value $m=\text{poly}(n/\varepsilon)$, we obtain $\Pr[h_{\mathbf{v}}\neq f]\leq O\left(\sqrt{\eta\log m}\right)$. Rearranging this bound, we can learn to accuracy ε even for $\eta=\Omega(\varepsilon^2/\log(n/\varepsilon))$. This completes the proof of the theorem. ■

4. Isotropic Log-concave Distributions and Malicious Noise

Our algorithm A_{mlc} that works for arbitrary isotropic log-concave distributions uses smooth boosting.

4.1 Smooth Boosting

A boosting algorithm uses a subroutine, called a *weak learner*, that is only guaranteed to output hypotheses with a non-negligible advantage over random guessing.[2] The boosting algorithm that we consider uses a *confidence-rated* weak learner (Schapire and Singer, 1999), which predicts $\{-1,1\}$ labels using continuous values in $[-1,1]$. Formally, the *advantage* of a hypothesis h' with respect to a distribution $\mathcal{D}'$ is defined to be $\mathbf{E}_{x\sim\mathcal{D}'}[h'(x)f(x)]$, where f is the target function.

For the purposes of this paper, a boosting algorithm makes use of the weak learner, an example oracle (possibly corrupted with noise), a desired accuracy ε, and a bound γ on the advantage of the hypothesis output by the weak learner.

A boosting algorithm that is trying to learn an unknown target function f with respect to some distribution $\mathcal{D}$ repeatedly simulates a (possibly noisy) example oracle for f with respect to some other distribution $\mathcal{D}'$ and calls a subroutine A_{weak} with respect to this oracle, receiving a *weak hypothesis*, which maps $\mathbf{R}^n$ to the continuous interval $[-1,1]$.

After repeating this for some number of stages, the boosting algorithm combines the weak hypotheses generated during its various calls to the weak learner into a final aggregate hypothesis which it outputs.

Let $\mathcal{D},\mathcal{D}'$ be two distributions over $\mathbf{R}^n$. We say that $\mathcal{D}'$ is $(1/\varepsilon)$-*smooth with respect to* $\mathcal{D}$ if $\mathcal{D}'(E)\leq(1/\varepsilon)\mathcal{D}(E)$ for all events E.

The following lemma from Servedio (2003) (similar results can be readily found elsewhere, see, e.g., Gavinsky 2003) identifies the properties that we need from a boosting algorithm for our analysis.

Lemma 13 (Servedio 2003) *There is a boosting algorithm B and a polynomial p such that, for any $\varepsilon,\gamma>0$, the following properties hold. When learning a target function f using $\mathrm{EX}_\eta(f,\mathcal{D})$, we have: (a) If each call to A_{weak} takes time t, then B takes time $p(t,1/\gamma,1/\varepsilon)$. (b) The weak learner is always called with an oracle $\mathrm{EX}_{\eta'}(f,\mathcal{D}')$ where $\mathcal{D}'$ is $(1/\varepsilon)$-smooth with respect to $\mathcal{D}$ and $\eta'\leq\eta/\varepsilon$. (c) Suppose that for each distribution $\mathrm{EX}_{\eta'}(f,\mathcal{D}')$ passed to A_{weak} by B, the output of A_{weak} has advantage γ. Then the final output h of B satisfies $\Pr_{x\in\mathcal{D}}[h(x)\neq f(x)]\leq\varepsilon$.*

4.2 The Algorithm

Our algorithm for learning under isotropic log-concave distributions with malicious noise, Algorithm A_{mlc}, applies the smooth booster from Lemma 13 with the following weak learner, which we call Algorithm A_{mlcw}. (The value c_0 is an absolute constant that will emerge from our analysis.)

2. For simplicity of presentation we ignore the confidence parameter of the weak learner in our discussion; this can be handled in an entirely standard way.

Algorithm A_{mlcw}:

1. Draw $m = \mathrm{poly}(n/\varepsilon)$ examples from the oracle $\mathrm{EX}_{\eta'}(f, \mathcal{D}')$.
2. Remove all those examples $(\mathbf{x}, y)$ for which $||\mathbf{x}|| > \sqrt{3n \log m}$.
3. Repeatedly
   - find a direction (unit vector) $\mathbf{w}$ that maximizes $\sum_{(\mathbf{x},y)\in S}(\mathbf{w}\cdot\mathbf{x})^2$ (see Lemma 6)
   - if $\sum_{(\mathbf{x},y)\in S}(\mathbf{w}\cdot\mathbf{x})^2 \leq c_0^2 m \log^2(n/\varepsilon)$ then move on to Step 4, and otherwise
   - remove from S all examples $(\mathbf{x}, y)$ for which $|\mathbf{w}\cdot\mathbf{x}| > c_0 \log(n/\varepsilon)$, and iterate again.
4. Let $\mathbf{v} = \frac{1}{|S|}\sum_{(\mathbf{x},y)\in S} y\mathbf{x}$, and return h defined by $h(\mathbf{x}) = \frac{\mathbf{v}\cdot\mathbf{x}}{3n\log m}$, if $|\mathbf{v}\cdot\mathbf{x}| \leq 3n\log m$, and $h(\mathbf{x}) = \mathrm{sgn}(\mathbf{v}\cdot\mathbf{x})$ otherwise.

4.3 The Key Claim: The Weak Learner is Effective

Our main task is to analyze the weak learner. Given the following Lemma, Theorem 2 will be an immediate consequence of Lemma 13.

Lemma 14 *Suppose Algorithm A_{mlcw} is run using $\mathrm{EX}_{\eta'}(f, \mathcal{D}')$ where f is an origin-centered halfspace, $\mathcal{D}'$ is $(1/\varepsilon)$-smooth w.r.t. an isotropic log-concave distribution $\mathcal{D}$, $\eta' \leq \eta/\varepsilon$, and $\eta \leq \Omega(\varepsilon^3/\log^2(n/\varepsilon))$. Then w.h.p. the hypothesis h returned by A_{mlcw} has advantage $\Omega\left(\frac{\varepsilon^2}{n\log(n/\varepsilon)}\right)$.*

Before proving Lemma 14, we need to prove some uniformity results on non-noisy examples drawn from an isotropic, log-concave distribution. This will enable us to use outlier removal and averaging to find a weak learner.

4.4 Lemmas in Support of Lemma 14

In this section, let us consider a single call to the weak learner with an oracle $EX_{\eta'}(f, \mathcal{D}')$ where $\mathcal{D}'$ is $(1/\varepsilon)$-smooth with respect to an isotropic log-concave distribution $\mathcal{D}$ and $\eta' \leq \eta/\varepsilon$. Our analysis will follow the same basic steps as Section 3.

A preliminary observation is that w.h.p. all clean examples drawn in Step 1 of Algorithm A_{mlcw} have $\|\mathbf{x}\| \leq \sqrt{3n\log m}$; indeed, for any given draw of $\mathbf{x}$ from $\mathcal{D}'$, the probability that $\|\mathbf{x}\| > \sqrt{3n\log m}$ is at most $\frac{e}{\varepsilon m^3}$ by Lemma 5 together with the fact that $\mathcal{D}'$ is $1/\varepsilon$-smooth with respect to an i.l.c. distribution. Therefore, w.h.p., only noisy examples are removed in Step 2 of the algorithm, and we shall assume that the distributions $\mathcal{D}$ and $\mathcal{D}'$ are in fact supported entirely on $\{\mathbf{x} : \|\mathbf{x}\| \leq \sqrt{3n\log m}\}$. This assumption affects us in two ways: first, it costs us an additional $\frac{e}{\varepsilon m^2}$ in the failure probability analysis below (which is not a problem and is in fact swallowed up by our "w.h.p." notation). Second, it means that the overall $1-\varepsilon$ accuracy bound we establish for the entire learning algorithm may be slightly worse than the true value. This is because our final hypothesis may always be wrong on the examples $\mathbf{x}$ that have $\|\mathbf{x}\| > \sqrt{3n\log m}$ and are ignored in our analysis; however such examples have probability mass at most $\frac{e}{m^3}$ under the isotropic log-concave distribution $\mathcal{D}$ (again by Lemma 5), and thus the additional accuracy cost is at most $\frac{e}{m^3}$. Since $\varepsilon \gg \frac{e}{m^3}$, this does not affect the overall correctness of our analysis. Note that a consequence of this assumption is that we can just take $h(\mathbf{x}) = \frac{\mathbf{v}\cdot\mathbf{x}}{3n\log m}$.

The remarks about high-probability statements and failure probabilities from Section 3.1 apply here as well, and as in Section 3 we write "w.h.p." as shorthand for "with probability $1 - 1/\mathrm{poly}(n/\varepsilon)$."

We first show that the variance of $\mathcal{D}'$ in every direction is not too large:

Lemma 15 *For any $\mathbf{a} \in \mathbb{S}^{n-1}$ we have $E_{\mathbf{x}\sim\mathcal{D}'}[(\mathbf{a}\cdot\mathbf{x})^2] = O(\log^2(1/\varepsilon))$.*

Proof. For $\mathbf{x}$ chosen according to $\mathcal{D}$, the distribution of $\mathbf{a}\cdot\mathbf{x}$ is a unit variance log-concave distribution by Lemma 4. Thus, for any positive integer k,

$$\begin{aligned} E_{\mathbf{x}\sim\mathcal{D}'}[(\mathbf{a}\cdot\mathbf{x})^2] &\leq k^2 + \sum_{i=k}^{\infty}(i+1)^2 \Pr_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \in (i,i+1]] \\ &\leq k^2 + \sum_{i=k}^{\infty}(i+1)^2(1/\varepsilon) \Pr_{\mathbf{x}\sim\mathcal{D}}[|\mathbf{a}\cdot\mathbf{x}| \in (i,i+1]] \\ &\leq k^2 + (1/\varepsilon)\sum_{i=k}^{\infty}(i+1)^2 \Pr_{\mathbf{x}\sim\mathcal{D}}[|\mathbf{a}\cdot\mathbf{x}| > i] \\ &\leq k^2 + (1/\varepsilon)\sum_{i=k}^{\infty}(i+1)^2 e^{-i+1} \leq k^2 + (1/\varepsilon)\cdot\Theta(k^2e^{-k}) \end{aligned}$$

where the first inequality in the last line uses Lemmas 4 and 5.

Setting $k = \ln(1/\varepsilon)$ completes the proof. ∎

The following anticoncentration bound will be useful for proving that clean examples drawn from $\mathcal{D}'$ tend to be classified correctly with a large margin.

Lemma 16 *Let $\mathbf{u} \in \mathbb{S}^{n-1}$. Then*

$$\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{u}\cdot\mathbf{x}|] \geq \varepsilon/8.$$

Proof. Clearly

$$\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{u}\cdot\mathbf{x}|] \geq (\varepsilon/4)\Pr_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{u}\cdot\mathbf{x}| > \varepsilon/4].$$

But by Lemma 5,

$$\Pr_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{u}\cdot\mathbf{x}| \leq \varepsilon/4] \leq \frac{1}{\varepsilon}\Pr_{\mathbf{x}\sim\mathcal{D}}[|\mathbf{u}\cdot\mathbf{x}| \leq \varepsilon/4] \leq \frac{\varepsilon/2}{\varepsilon} = 1/2.$$

∎

The next two lemmas are isotropic log-concave analogues of the uniform distribution Lemmas 7 and 8 respectively. The first one says that w.h.p. no direction $\mathbf{a}$ has much more variance than the expected variance in any direction:

Lemma 17 *W.h.p. over a random draw of ℓ clean examples S_{clean} from $\mathcal{D}'$, we have*

$$\max_{\mathbf{a}\in\mathbb{S}^{n-1}}\left\{\frac{1}{\ell}\sum_{(\mathbf{x},y)\in S_{\text{clean}}}(\mathbf{a}\cdot\mathbf{x})^2\right\} \leq O(1)\left(\log^2\frac{1}{\varepsilon} + \frac{n^{3/2}\log^2\ell}{\sqrt{\ell}}\right).$$

Proof. By Lemma 15, for any $\mathbf{a} \in \mathbb{S}^{n-1}$ we have

$$\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[(\mathbf{a}\cdot\mathbf{x})^2] = \Theta(\log^2(1/\varepsilon)).$$

Since as remarked earlier we may assume $\mathcal{D}'$ is supported on $\{\mathbf{x} : \|\mathbf{x}\| \leq \sqrt{3n\log m}\}$, we may apply Lemmas 25 and 27 (see Appendix A) with functions $f_{\mathbf{a}}$ defined by $f_{\mathbf{a}} = \frac{(\mathbf{a}\cdot\mathbf{x})^2}{3n\log m}$. This completes the proof. ∎

The second lemma says that for a sufficiently large clean data set, w.h.p. no direction has too many examples lying too far out in that direction:

Lemma 18 *For any $\beta > 0$ and $\kappa > 1$, if S_{clean} is a set of $\ell \geq \frac{O(1)\varepsilon e^{\beta}\left(n\ln\left(e^{-\beta}/\varepsilon\right)+\log m\right)}{(1+\kappa)\ln(1+\kappa)}$ clean examples drawn from $\mathcal{D}'$, then w.h.p. we have*

$$\max_{\mathbf{a}\in\mathbb{S}^{n-1}} \frac{1}{\ell}|\{\mathbf{x}\in S_{\text{clean}} : |\mathbf{a}\cdot\mathbf{x}| > \beta\}| \;\leq\; (1+\kappa)\left(\frac{1}{\varepsilon}\right)e^{-\beta+1}.$$

Proof. Lemma 5 implies that for the original isotropic log-concave distribution $\mathcal{D}$, we have

$$\Pr_{\mathbf{x}\sim\mathcal{D}}[|\mathbf{a}\cdot\mathbf{x}| > \beta] \leq e^{-\beta+1}.$$

Since $\mathcal{D}'$ is $(1/\varepsilon)$-smooth with respect to $\mathcal{D}$, this implies that

$$\Pr_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| > \beta] \leq \frac{e^{-\beta+1}}{\varepsilon}. \tag{5}$$

In the proof of Lemma 8, we observed that the VC-dimension of

$$\{\{\mathbf{x} : |\mathbf{a}\cdot\mathbf{x}| > \beta\} \;:\; \mathbf{a}\in\mathbf{R}^n, \beta\in\mathbf{R}\}$$

is $O(n)$, so applying Lemma 28 with (5) completes the proof of this lemma. ∎

The following is an isotropic log-concave analogue of Corollary 9, establishing that not too many clean examples are removed in the outlier removal step:

Corollary 19 *W.h.p. over the random draw of the m-element sample S from $EX_{\eta'}(f,\mathcal{D}')$, the number of clean examples removed during any one execution of the outlier removal step (final substep of Step 2) in Algorithm A_{mlcw} is at most $6m\varepsilon^3/n^4$.*

Proof. Since the true noise rate η is assumed sufficiently small, the value $\eta' \leq \eta/\varepsilon$ is at most $\varepsilon/4$, and thus w.h.p. the number ℓ of clean examples in S is at least (say) $m/2$. We would like to apply Lemma 18 with $\kappa = (n/\varepsilon)^{c_0-4}$ and $\beta = c_0\log(n/\varepsilon)$, and we may do this since we have

$$\frac{O(1)\varepsilon e^{\beta}\left(n\ln\left(\varepsilon e^{\beta}\right)+\log m\right)}{(1+\kappa)\ln(1+\kappa)} \leq \frac{O(1)\varepsilon(n/\varepsilon)^{c_0}n\log m}{(n/\varepsilon)^{c_0-4}\log m} \leq O(1)n^5/\varepsilon^3 \ll \frac{m}{2} \leq \ell$$

for a suitable fixed $\text{poly}(n/\varepsilon)$ choice of m. Since clean points are only removed if they have $|\mathbf{a}\cdot\mathbf{x}| \geq \beta$, Lemma 18 gives us that the number of clean points removed is at most

$$m(1+\kappa)\cdot\frac{1}{\varepsilon}e^{-\beta+1} \leq m\frac{(6/\varepsilon)(n/\varepsilon)^{c_0-4}}{(n/\varepsilon)^{c_0}} \leq 6m\varepsilon^3/n^4.$$

∎

The following lemma is an analogue of Lemma 10; it lower bounds the number of dirty examples that are removed in the outlier removal step.

Lemma 20 *W.h.p. over the random draw of S, any time Algorithm A_{mlcw} executes the outlier removal step it removes at least $\frac{m}{O(n)}$ noisy examples.*

Proof. Since our ultimate goal is only to prove that the algorithm succeeds for some η which is $o(\varepsilon)$, we may assume without loss of generality that the original noise rate η is less than $\varepsilon/4$. This means that $\eta' < 1/4$, and consequently a Chernoff bound gives that w.h.p. the fraction $\widehat{\eta}'$ of noisy examples in S at the beginning of the weak learner's training is at most $1/2$. And Lemma 17 implies that for a sufficiently large polynomial choice of m, we have that w.h.p. for all $\mathbf{a} \in \mathbb{S}^{n-1}$, the following holds for all the clean examples in the data before any examples were removed:

$$\sum_{(\mathbf{x},y)\in S_{\mathrm{clean}}} (\mathbf{a}\cdot\mathbf{x})^2 \leq cm\log^2(1/\varepsilon) \tag{6}$$

where c is an absolute constant. We say that a random sample that meets all these requirements is "reasonable." We now set the constant c_0 that is used in the specification of A_{mlcw} to be $\sqrt{2(c+1)}$. We will now show that, for any reasonable sample S, the number of noisy examples removed during the first execution of the outlier removal step of A_{mlcw} is at least $\frac{m}{O(n)}$.

If we remove examples using direction $\mathbf{w}$ then it means $\sum_{x\in S}(\mathbf{w}\cdot\mathbf{x})^2 \geq c_0^2 m\log^2(n/\varepsilon)$. Since S is reasonable, by (6) the contribution to the sum from the clean examples that have survived until this point is at most $cm\log^2(1/\varepsilon)$ so we must have

$$\sum_{(\mathbf{x},y)\in S_{\mathrm{dirty}}} (\mathbf{w}\cdot\mathbf{x})^2 \geq (c_0^2 - c)m\log^2(n/\varepsilon).$$

Let $S_{\mathrm{dirty}} = N \cup F$ where N is the examples $(\mathbf{x},y)$ for which $\mathbf{x}$ satisfies $(\mathbf{w}\cdot\mathbf{x})^2 \leq c_0^2\log^2(n/\varepsilon)$ and F is the other points. We have

$$\sum_{(\mathbf{x},y)\in N} (\mathbf{w}\cdot\mathbf{x})^2 \leq c_0^2\widehat{\eta}' m\log^2(n/\varepsilon).$$

and so, since $||\mathbf{x}|| \leq \sqrt{3n\log m}$ implies that $(\mathbf{w}\cdot\mathbf{x})^2 \leq 3n\log m$ for all unit length $\mathbf{w}$, we have

$$\begin{aligned}
|F| &\geq \sum_{(\mathbf{x},y)\in F} \frac{(\mathbf{w}\cdot\mathbf{x})^2}{3n\log m} = \sum_{(\mathbf{x},y)\in S_{\mathrm{dirty}}} \frac{(\mathbf{w}\cdot\mathbf{x})^2}{3n\log m} - \sum_{(\mathbf{x},y)\in N} \frac{(\mathbf{w}\cdot\mathbf{x})^2}{3n\log m} \\
&\geq \frac{(c_0^2 - c)m\log^2(n/\varepsilon) - c_0^2\widehat{\eta}' m\log^2(n/\varepsilon)}{3n\log m} \\
&\geq \frac{m\log^2(n/\varepsilon)}{3n\log m} \\
&\geq \frac{m}{O(n)}
\end{aligned}$$

where the next-to-last inequality uses $\eta' \leq 1/2$ and $c_0 = \sqrt{2(c+1)}$, and the final one uses $m = O(\mathrm{poly}(n/\varepsilon))$. The points in F are precisely the ones that are removed, and thus the lemma is proved. ∎

4.5 Proof of Lemma 14

We first note that Lemma 20 implies that w.h.p. the weak learner must terminate after at most $O(n)$ iterations of outlier removal.

Let $\mathbf{u}$ be the unit length normal vector of the separating halfspace for the target function f. Recall that we have assumed without loss of generality that $||\mathbf{x}|| \leq \sqrt{3n\log m}$ for all $\mathbf{x}$ in the training set, so that $||\mathbf{v}|| \leq \sqrt{3n\log m}$, and thus the advantage of h with respect to $\mathcal{D}'$ can be expressed as

$$\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[h(\mathbf{x})f(\mathbf{x})] = \frac{\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[(\mathbf{v}\cdot\mathbf{x})f(\mathbf{x})]}{3n\log m} \tag{7}$$

and so we shall work on lower bounding $\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[(\mathbf{v}\cdot\mathbf{x})f(\mathbf{x})]$.

As in the proof of Lemma 12, let

- S_{clean} be all of the clean examples in the initial sample S, and S'_{clean} be those that are not removed in any stage of outlier removal;
- S_{dirty} be all of the dirty examples in the initial sample S, and S'_{dirty} be those that are not removed in any stage of outlier removal;
- $\eta' = \frac{|S'_{\text{dirty}}|}{|S'_{\text{clean}}\cup S'_{\text{dirty}}|}$, that is, the noise rate among the examples that survive until the end of training of the weak learner, and
- $\alpha = \frac{|S_{\text{clean}} - S'_{\text{clean}}|}{|S'_{\text{clean}}\cup S'_{\text{dirty}}|}$, the ratio of the number of clean points that were erroneously removed to the size of the final surviving data set.

As before we write S' for $S'_{\text{clean}} \cup S'_{\text{dirty}}$. Also as before, let $\mathbf{v}_{\text{clean}}$ be the average of *all* the clean examples, $\mathbf{v}'_{\text{dirty}}$ be the average of the dirty (noisy) examples that were not deleted, and $\mathbf{v}_{\text{del}}$ be the average of the clean examples that were deleted. Then arguing exactly as before, we have

$$\mathbf{v} = (1-\eta'+\alpha)\mathbf{v}_{\text{clean}} + \eta'\mathbf{v}'_{\text{dirty}} - \alpha\mathbf{v}_{\text{del}}.$$

The expectation of $\mathbf{v}_{\text{clean}}$ will play a special role in the analysis:

$$\mathbf{v}^*_{\text{clean}} \stackrel{def}{=} \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})\mathbf{x}].$$

Once again, we will demonstrate the limited effect of $\mathbf{v}'_{\text{dirty}}$ by bounding its length. This time, the outlier removal enforces the fact that, for any $\mathbf{w} \in \mathbb{S}^{n-1}$, we have

$$\sum_{(\mathbf{x},y)\in S} (\mathbf{w}\cdot\mathbf{x})^2 \leq c_0^2 m\log^2(n/\varepsilon).$$

Applying this for the unit vector $\mathbf{w}$ in the direction of $\mathbf{v}'_{\text{dirty}}$ as was done in Lemma 12, this implies

$$\|\mathbf{v}'_{\text{dirty}}\| \leq c_0\log(n/\varepsilon)\sqrt{\frac{m}{|S'_{\text{dirty}}|}}.$$

Next, let us apply this to bound an expression that captures the average harm done by $\mathbf{v}'_{\text{dirty}}$.

$$\begin{aligned} |\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})(\mathbf{v}'_{\text{dirty}}\cdot\mathbf{x})]| &= |\mathbf{v}'_{\text{dirty}}\cdot\mathbf{v}^*_{\text{clean}}| \\ &\leq c_0\log(n/\varepsilon)\sqrt{\frac{m}{|S'_{\text{dirty}}|}}||\mathbf{v}^*_{\text{clean}}||. \end{aligned} \tag{8}$$

To show that $\mathbf{v}_{\text{clean}}$ plays a relatively large role, it is helpful to lower bound the length of $\mathbf{v}^*_{\text{clean}}$. We do this by lower bounding the length of its projection onto the unit normal vector $\mathbf{u}$ of the target as follows:

$$\mathbf{v}^*_{\text{clean}}\cdot\mathbf{u} = \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[(f(\mathbf{x})\mathbf{x})\cdot\mathbf{u}] = \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[\text{sgn}(\mathbf{u}\cdot\mathbf{x})(\mathbf{x}\cdot\mathbf{u})] = \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{x}\cdot\mathbf{u}|] \geq \varepsilon/8,$$

by Lemma 16. Since $\mathbf{u}$ is unit length, this implies

$$||\mathbf{v}^*_{\text{clean}}|| \geq \varepsilon/8. \tag{9}$$

Armed with this bound, we can now lower bound the benefit imparted by $\mathbf{v}_{\text{clean}}$:

$$\begin{aligned} \mathbf{E}_{\mathbf{z}\sim\mathcal{D}'}[f(\mathbf{z})(\mathbf{v}_{\text{clean}}\cdot\mathbf{z})] &= \frac{1}{S_{\text{clean}}}\sum_{(\mathbf{x},y)\in S_{\text{clean}}}\mathbf{E}_{\mathbf{z}\sim\mathcal{D}'}[yf(\mathbf{z})(\mathbf{x}\cdot\mathbf{z})] \\ &= \frac{1}{S_{\text{clean}}}\sum_{(\mathbf{x},y)\in S_{\text{clean}}}(y\mathbf{x})\cdot\mathbf{v}^*_{\text{clean}}. \end{aligned}$$

Since $\mathbf{E}[(y\mathbf{x})\cdot\mathbf{v}^*_{\text{clean}}] = ||\mathbf{v}^*_{\text{clean}}||^2$, and $(y\mathbf{x})\cdot\mathbf{v}^*_{\text{clean}} \in [-3n\log m, 3n\log m]$, a Hoeffding bound implies that w.h.p.

$$\mathbf{E}_{\mathbf{z}\sim\mathcal{D}'}[f(\mathbf{z})(\mathbf{v}_{\text{clean}}\cdot\mathbf{z})] \geq ||\mathbf{v}^*_{\text{clean}}||^2 - O(n\log^{3/2} m)/\sqrt{|S_{\text{clean}}|}.$$

Since the noise rate η' is at most η/ε and η certainly less than $\varepsilon/4$ as discussed above, another Hoeffding bound gives that w.h.p. $|S_{\text{clean}}|$ is at least $m/2$; thus for a suitably large polynomial choice of m, using (9) we have

$$\mathbf{E}_{\mathbf{z}\sim\mathcal{D}'}[f(\mathbf{z})(\mathbf{v}_{\text{clean}}\cdot\mathbf{z})] \geq ||\mathbf{v}^*_{\text{clean}}||^2 - O(n\log^{3/2} m)/\sqrt{m/2} \geq \frac{||\mathbf{v}^*_{\text{clean}}||^2}{2}. \tag{10}$$

Now we are ready to put our bounds together and lower bound the advantage of $\mathbf{v}$. We have

$$\begin{aligned} \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})(\mathbf{v}\cdot\mathbf{x})] &= (1-\eta'+\alpha)\mathbf{E}[f(\mathbf{x})(\mathbf{v}_{\text{clean}}\cdot\mathbf{x})] \\ &\quad +\eta'\mathbf{E}[f(\mathbf{x})(\mathbf{v}'_{\text{dirty}}\cdot\mathbf{x})] - \alpha\mathbf{E}[f(\mathbf{x})(\mathbf{v}_{\text{del}}\cdot\mathbf{x})]. \end{aligned}$$

We bound each of the three contributions in turn. First, using $1-\eta' \geq 1/2$ and (10), we have $(1-\eta'+\alpha)\mathbf{E}[f(\mathbf{x})(\mathbf{v}_{\text{clean}}\cdot\mathbf{x})] \geq \frac{||\mathbf{v}^*_{\text{clean}}||^2}{4}$.

Next, by (8), we have

$$|\eta'\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})(\mathbf{v}'_{\text{dirty}}\cdot\mathbf{x})]| \leq c_0\log(n/\varepsilon)\sqrt{2\eta'}||\mathbf{v}^*_{\text{clean}}||.$$

Since we may assume that $\eta \leq c'\varepsilon^3/\log^2(n/\varepsilon)$ for as small a fixed constant c' as we like (recall the overall bound of Theorem 2), we get

$$c_0\log(n/\varepsilon)\sqrt{2\eta'}||\mathbf{v}^*_{\text{clean}}|| \leq (\varepsilon/64)||\mathbf{v}^*_{\text{clean}}||$$

(for a suitably small constant choice of c'), and this is less than $\frac{||\mathbf{v}^*_{\text{clean}}||^2}{8}$ since $||\mathbf{v}^*_{\text{clean}}|| \geq \varepsilon/8$.

Finally Corollary 19, together with the fact that there are at most $O(n)$ iterations of outlier removal and the final surviving data set is of size at least $m/4$, gives us that $\alpha \leq \frac{O(n)(6m\varepsilon^3/n^4)}{m/4}$, which (recalling that both $\mathbf{v}_{\text{del}}$ and all $\mathbf{x}$ in the support of $\mathcal{D}'$ have norm at most $\sqrt{3n\log m}$) means that $|\alpha \mathbf{E}[f(\mathbf{x})(\mathbf{v}_{\text{del}} \cdot \mathbf{x})]| = o(\varepsilon^2)$.

Combining all these bounds, we get

$$\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})(\mathbf{v}\cdot\mathbf{x})] \geq \frac{||\mathbf{v}^*_{\text{clean}}||^2}{4} - \frac{||\mathbf{v}^*_{\text{clean}}||^2}{8} - o(\varepsilon^2) \geq \frac{\varepsilon^2}{1024}$$

by (9). Together with (7), the proof of Lemma 14 is completed.

5. Learning Under Isotropic Log-concave Distributions with Adversarial Label Noise

In this section, we consider the model where an adversary can change some class labels, but cannot otherwise modify examples.

5.1 The Model

We now define the model of learning with adversarial label noise under isotropic log-concave distributions. In this model the learning algorithm has access to an oracle that provides independent random examples drawn according to a fixed distribution P on $\mathbf{R}^n \times \{-1,1\}$, where

- the marginal distribution over $\mathbf{R}^n$ is isotropic log-concave, and
- there is a halfspace f such that $\Pr_{(\mathbf{x},y)\sim P}[f(\mathbf{x}) \neq y] = \eta$.

The parameter η is the *noise rate*. As usual, the goal of the learner is to output a hypothesis h such that $\Pr_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x}) \neq y] \leq \varepsilon$; if an algorithm achieves this goal, we say it learns to accuracy $1-\varepsilon$ in the presence of adversarial label noise at rate η.

5.2 The Algorithm

Like the algorithm A_{mlc} considered in the last section, the algorithm A_{alc} studied in this section applies the smooth boosting algorithm of Lemma 13 to a weak learner that performs averaging. The weak learner A_{alcw} behaves as follows:

Algorithm A_{alcw}:

1. Draw a set S of m examples according to P' (the oracle for a modified distribution provided by the boosting algorithm).
2. Remove all examples $(\mathbf{x},y)$ such that $||\mathbf{x}|| > \sqrt{3n\log m}$ from S.
3. Let $\mathbf{v} = \frac{1}{|S|}\sum_{(\mathbf{x},y)\in S} y\mathbf{x}$. Return the confidence-rated classifier h defined by $h(\mathbf{x}) = \frac{\mathbf{v}\cdot\mathbf{x}}{3n\log m}$ if $|\mathbf{v}\cdot\mathbf{x}| \leq 3n\log m$, and $h(\mathbf{x}) = \text{sgn}(\mathbf{v}\cdot\mathbf{x})$ otherwise.

5.3 Claim About the Weak Learner

As in the previous section, the heart of our analysis will be to analyze the weak learner. We omit discussing the application of the smooth boosting algorithm here, as it is nearly identical to Section 4.

Lemma 21 *Suppose Algorithm A_{alcw} is run using P' as the source of labeled examples, where P' is a distribution that is $(1/\varepsilon)$-smooth with respect to a joint distribution P on $\mathbf{R}^n \times \{-1,1\}$ whose marginal $\mathcal{D}'$ on $\mathbf{R}^n$ is isotropic and log-concave. Further, assume there exists a linear threshold function f such that $\Pr_{(\mathbf{x},y)\sim P'}[f(\mathbf{x}) \neq y] \leq \eta/\varepsilon$ and $\eta \leq \Omega(\frac{\varepsilon^3}{\log(1/\varepsilon)})$. Then with high probability, A_{alcw} outputs a hypothesis with advantage $\Omega(\frac{\varepsilon^2}{n\log(n/\varepsilon)})$.*

5.4 Lemmas in Support of Lemma 21

During this section, let us focus our attention on a single call to the weak learner. Let P' be a distribution as in Lemma 21 and let $\mathcal{D}'$ be the marginal on $\mathbf{R}^n$. We observe that since P' is $(1/\varepsilon)$-smooth with respect to P, the marginal $\mathcal{D}'$ of P' is $(1/\varepsilon)$-smooth with respect to the marginal $\mathcal{D}$ of P.

As in Section 4, we may assume that the support of $\mathcal{D}'$ lies entirely on $\mathbf{x}$ such that $||\mathbf{x}|| \leq \sqrt{3n\log m}$ (this negligibly affects the final bounds obtained in our analyses).

The following technical lemma will be used to limit the extent to which the distribution P' can concentrate a lot of noise in one direction.

Lemma 22 *Let E be any event with positive probability under $\mathcal{D}'$, and let $\kappa = \mathcal{D}'(E)$. For any unit length $\mathbf{a} \in \mathbf{R}^n$, $\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid E] = O\left(\log \frac{1}{\kappa\varepsilon}\right)$.*

Proof. Let β be such that $\Pr_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| > \beta] = \kappa$. By Lemmas 4 and 5, together with the fact that $\mathcal{D}'$ is $(1/\varepsilon)$ smooth with respect to $\mathcal{D}$, we have

$$\kappa \leq \frac{1}{\varepsilon} e^{-\beta+1}$$

which implies $\beta \leq 1 + \ln\left(\frac{1}{\varepsilon\kappa}\right)$.

Let F be the event that $|\mathbf{a}\cdot\mathbf{x}| > \beta$. We will show that $\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid E] \leq \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid F]$, and then bound $\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid F]$. If $\Pr[(E-F)\cup(F-E)] = 0$, then, obviously, $\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid E] = \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid F]$. Suppose $\Pr[(E-F)\cup(F-E)] > 0$. Then

$$\begin{aligned}
&\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid E] \\
&= \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid E\cap F]\Pr[E\cap F] + \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid E-F]\Pr[E-F] \\
&= \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid E\cap F]\Pr[E\cap F] + \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid E-F]\Pr[F-E] \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{(because } \Pr[E] = \Pr[F]\text{)} \\
&< \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid E\cap F]\Pr[E\cap F] + \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid F-E]\Pr[F-E],
\end{aligned}$$

because for every $\mathbf{x} \in E-F$ and every $\mathbf{x}' \in F-E$,

$$|\mathbf{a}\cdot\mathbf{x}| \leq \beta < |\mathbf{a}\cdot\mathbf{x}'|.$$

But

$$\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid E\cap F]\Pr[E\cap F] + \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid F-E]\Pr[F-E] = \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid F],$$

so

$$\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid E] < \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid F]. \tag{11}$$

Now, setting $b = \lfloor \beta \rfloor$, we have

$$\begin{aligned}
\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \mid F] &\leq \frac{1}{\mathcal{D}'(F)} \sum_{i=b} (i+1) \Pr_{\mathbf{x}\sim\mathcal{D}'}[|\mathbf{a}\cdot\mathbf{x}| \in (i, i+1]] \\
&\leq \frac{1}{\mathcal{D}'(F)} \sum_{i=b} (i+1) e^{-i+1} \\
&= \frac{1}{\mathcal{D}'(F)} \left(O\left(\frac{e^{-b} b}{\varepsilon} \right) \right) \\
&= O(b),
\end{aligned}$$

since $\mathcal{D}'(F) = \Theta(e^{-b}/\varepsilon)$. Combining with (11) completes the proof. ∎

5.5 Proof of Lemma 21

Fix some halfspace f such that $\Pr_{(\mathbf{x},y)\sim P}[f(\mathbf{x}) \neq y] = \eta$, and let $\mathbf{u}$ be the unit normal vector of its separating hyperplane.

Let P' be the joint distribution given to A_{alcw} and let $\mathcal{D}'$ be its marginal on $\mathbf{R}^n$. As noted in the previous subsection, $\mathcal{D}'$ is $(1/\varepsilon)$-smooth with respect to the original marginal distribution $\mathcal{D}$ of P.

First, we bound the advantage of the hypothesis h with respect to P' in terms of the tendency of h to agree with the best linear function f:

$$\mathbf{E}_{(\mathbf{x},y)\sim P'}[h(\mathbf{x})y] \geq \mathbf{E}_{(\mathbf{x},y)\sim P'}[h(\mathbf{x})f(\mathbf{x})] - \eta = \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[h(\mathbf{x})f(\mathbf{x})] - \eta. \tag{12}$$

Furthermore, as we have assumed without loss of generality that $||\mathbf{x}|| \leq \sqrt{3n\log m}$ for all examples in the training set, and therefore that $||\mathbf{v}|| \leq \sqrt{3n\log m}$, we have

$$\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[h(\mathbf{x})f(\mathbf{x})] = \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}\left[\frac{f(\mathbf{x})(\mathbf{x}\cdot\mathbf{v})}{3n\log m} \right] \tag{13}$$

so we will work on bounding $\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})(\mathbf{x}\cdot\mathbf{v})]$.

Let P'_{clean} be obtained by conditioning a random draw $(\mathbf{x}, y)$ from P' on the event that $f(\mathbf{x}) = y$. Define P'_{dirty} analogously, and let $\mathcal{D}'_{\text{clean}}$ and $\mathcal{D}'_{\text{dirty}}$ be the corresponding marginals on $\mathbf{R}^n$. Let

$$\begin{aligned}
\mathbf{v}^*_{\text{dirty}} &= \mathbf{E}_{(\mathbf{x},y)\sim P'_{\text{dirty}}}[y\mathbf{x}] \\
\mathbf{v}^*_{\text{correct}} &= \mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})\mathbf{x}].
\end{aligned}$$

Note that the linearity of expectation implies that

$$\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})(\mathbf{x}\cdot\mathbf{v})] = (\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})(\mathbf{x})])\cdot\mathbf{v} = \mathbf{v}^*_{\text{correct}}\cdot\mathbf{v} = \frac{1}{m}\sum_{(\mathbf{x},y)\in S} \mathbf{v}^*_{\text{correct}}\cdot(y\mathbf{x}). \tag{14}$$

Equation (14) expresses $\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})(\mathbf{x}\cdot\mathbf{v})]$, which is closely related to the advantage of h through (13) and (12), as a sum of independent random variables, one for each example. We will bound $\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})(\mathbf{x}\cdot\mathbf{v})]$ by bounding the expected effect of a random example on its value, and applying a Hoeffding bound.

Let $\eta' = \Pr_{(\mathbf{x},y)\sim P'}[f(\mathbf{x}) \neq y]$. Since P' is $1/\varepsilon$-smooth with respect to P, we have $\eta' \leq \eta/\varepsilon$. We can rearrange the effect of a random example as follows

$$\begin{aligned}
\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (y\mathbf{x})] &= (1-\eta')\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})|y = f(\mathbf{x})] \\
&\quad +\eta'\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (-f(\mathbf{x})\mathbf{x})|y \neq f(\mathbf{x})] \\
&= (1-\eta')\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})|y = f(\mathbf{x})] \\
&\quad +\eta'\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})|y \neq f(\mathbf{x})] \\
&\quad -\eta'\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})|y \neq f(\mathbf{x})] \\
&\quad +\eta'\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (-f(\mathbf{x})\mathbf{x})|y \neq f(\mathbf{x})]. \qquad (15)
\end{aligned}$$

Since

$$\begin{aligned}
&\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})] \\
&= \eta'\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})|y \neq f(\mathbf{x})] + (1-\eta')\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})|y = f(\mathbf{x})],
\end{aligned}$$

by replacing the first two terms of (15) with $\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})]$, we get

$$\begin{aligned}
\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (y\mathbf{x})] &= \mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})] \\
&\quad -\eta'\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})|y \neq f(\mathbf{x})] \\
&\quad +\eta'\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (-f(\mathbf{x})\mathbf{x})|y \neq f(\mathbf{x})] \\
&= \mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})] \\
&\quad -2\eta'\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})|y \neq f(\mathbf{x})].
\end{aligned}$$

Twice applying the linearity of expectation, we get

$$\begin{aligned}
\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (y\mathbf{x})] &= ||\mathbf{v}^*_{\text{correct}}||^2 - 2\eta'\mathbf{E}_{(\mathbf{x},y)\sim P'}[\mathbf{v}^*_{\text{correct}} \cdot (f(\mathbf{x})\mathbf{x})|y \neq f(\mathbf{x})] \\
&= ||\mathbf{v}^*_{\text{correct}}||^2 - 2\eta'\mathbf{v}_{\text{correct}} \cdot \mathbf{v}^*_{\text{dirty}} \\
&\geq ||\mathbf{v}^*_{\text{correct}}||^2 - 2\eta'||\mathbf{v}^*_{\text{correct}}|| \cdot ||\mathbf{v}^*_{\text{dirty}}|| \\
&\geq \frac{1}{2}||\mathbf{v}^*_{\text{correct}}||^2 - 4(\eta')^2||\mathbf{v}^*_{\text{dirty}}||^2,
\end{aligned}$$

The last line follows from the fact that $q^2 - qr \geq (q^2 - r^2)/2$ for all real q, r.

So now our goals are a lower bound on $||\mathbf{v}^*_{\text{correct}}||$ and an upper bound on $||\mathbf{v}^*_{\text{dirty}}||$.

We can lower bound $||\mathbf{v}^*_{\text{correct}}||$ essentially the same way we did before, by lower bounding its projection onto the "target" normal vector $\mathbf{u}$:

$$\mathbf{v}^*_{\text{correct}} \cdot \mathbf{u} = \mathbf{E}_{(\mathbf{x},y)\sim P'}[(f(\mathbf{x})\mathbf{x}) \cdot \mathbf{u}] = \mathbf{E}_{(\mathbf{x},y)\sim P'}[\text{sgn}(\mathbf{u} \cdot \mathbf{x})(\mathbf{x} \cdot \mathbf{u})] = \mathbf{E}_{(\mathbf{x},y)\sim P'}[|\mathbf{x} \cdot \mathbf{u}|] \geq \varepsilon/16, \qquad (16)$$

by Lemma 16.

We upper bound $||\mathbf{v}^*_{\text{dirty}}||$ as follows:

$$\begin{aligned}
||\mathbf{v}^*_{\text{dirty}}||^2 &= \mathbf{v}^*_{\text{dirty}} \cdot E_{\mathbf{x}\sim\mathcal{D}'_{\text{dirty}}}[-f(\mathbf{x})\mathbf{x}] \\
&= ||\mathbf{v}^*_{\text{dirty}}|| \cdot E_{\mathbf{x}\sim\mathcal{D}'_{\text{dirty}}}\left[\left(\frac{\mathbf{v}^*_{\text{dirty}}}{||\mathbf{v}^*_{\text{dirty}}||}\right) \cdot (-f(\mathbf{x}))\mathbf{x}\right] \\
&\leq ||\mathbf{v}^*_{\text{dirty}}|| \cdot E_{\mathbf{x}\sim\mathcal{D}'_{\text{dirty}}}\left[\left|\left(\frac{\mathbf{v}^*_{\text{dirty}}}{||\mathbf{v}^*_{\text{dirty}}||}\right) \cdot \mathbf{x}\right|\right] \\
&\leq ||\mathbf{v}^*_{\text{dirty}}||O(\log(1/(\eta'\varepsilon)))
\end{aligned}$$

by Lemma 22. Thus $||\mathbf{v}^*_{\text{dirty}}|| \leq O(\log(1/(\eta'\varepsilon)))$.

Combining this with (16) and (14) we have that if

$$\eta'\sqrt{\log(1/(\eta'\varepsilon)} \leq c\varepsilon^2$$

for a suitably small constant c, then $\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})(\mathbf{x}\cdot\mathbf{v})]$ is a sum of m i.i.d. random variables, each with mean at least $\Omega(\varepsilon^2)$, and coming from an interval of length $O(n\log m)$. Applying the standard Hoeffding bound, polynomially many examples suffice for $\mathbf{E}_{\mathbf{x}\sim\mathcal{D}'}[f(\mathbf{x})(\mathbf{x}\cdot\mathbf{v})] \geq \Omega(\varepsilon^2)$. Combining with (13) and (12) completes the proof.

6. Conclusion

Our algorithms use boosting together with a confidence-rated weak learner that perform a simple averaging of labeled examples. As shown in earlier work (Servedio, 2002, 2003) there are close connections between such an approach and the Perceptron algorithm. It seems likely that the Perceptron could be used as an alternative to boosting and averaging in our algorithms; it would be interesting to see if a Perceptron-based approach has any theoretical or empirical advantages over the algorithms we give in this paper.

More generally, there are relatively few algorithms for learning interesting classes of functions in the presence of malicious noise. We hope that our results will help lead to the development of more efficient algorithms for this challenging noise model.

As a challenge for future work, we pose the following question: do there exist computationally efficient algorithms for learning halfspaces under *arbitrary* distributions in the presence of malicious noise? As of now no better results are known for this problem than the generic conversions of Kearns and Li (1993), which can be applied to any concept class. We feel that even a small improvement in the malicious noise rate that can be handled for halfspaces would be a very interesting result.

Acknowledgments

We are grateful to the anonymous reviewers for their comments.

Appendix A. Proof of Lemma 7

Let us start with a couple of definitions and a couple of bounds from the literature.

Definition 23 (VC-dimension) *A set F of $\{-1,1\}$-valued functions defined on a common domain X* shatters *$x_1,...,x_d$ if every sequence $y_1,...,y_d \in \{-1,1\}$ of function values has a function f such that $f(x_1) = y_1,...,f(x_d) = y_d$. The VC-dimension of F is the size of the largest set shattered by F.*

Definition 24 (pseudo-dimension) *For a set F of real-valued functions defined on a common domain X, the* pseudo-dimension *of F is the VC-dimension of $\{\text{sign}(f(\cdot)-\theta) : f \in F, \theta \in \mathbf{R}\}$.*

Lemma 25 (Pollard 1984; Talagrand 1994) *Let F be a set of real-valued functions defined on a common domain X taking values in $[0,1]$, and let d be the pseudo-dimension of F. Let $\mathcal{D}$ be a probability distribution over X. Then if $x_1,...,x_m$ are obtained by drawing m times independently*

according to $\mathcal{D}$, for any $\delta > 0$,

$$\Pr\left[\exists f \in F, \frac{1}{m}\sum_{s=1}^{m} f(x_s) > E_{\mathcal{D}}[f] + c\sqrt{\frac{d+\log(1/\delta)}{m}}\right] \leq \delta,$$

where $c > 0$ is an absolute constant.

Lemma 26 (see Blumer et al. 1989) *The VC-dimension of unions of two halfspaces is $O(n)$.*

Now, let us bound the pseudo-dimension of the class of functions that we need.

Lemma 27 *Let F_n consist of the functions f from $\mathbf{R}^n$ to $\mathbf{R}$ which can be defined by $f(\mathbf{x}) = (\mathbf{a}\cdot\mathbf{x})^2$ for some $\mathbf{a} \in \mathbf{R}^n$. The pseudo-dimension of F_n is at most $O(n)$.*

Proof. According to the definition, the pseudo dimension of F_n is the VC-dimension of the set G_n of $\{-1,1\}$-valued functions $g_{\mathbf{a},\theta}$ defined by $g_{\mathbf{a},\theta}(\mathbf{x}) = \mathrm{sign}((\mathbf{a}\cdot\mathbf{x})^2 - \theta)$. Each $g_{\mathbf{a},\theta}$ is equivalent to an OR of two halfspaces:

$$\mathbf{a}\cdot\mathbf{x} \geq \sqrt{\theta} \quad \text{OR} \quad (-\mathbf{a})\cdot\mathbf{x} \geq \sqrt{\theta}.$$

Thus the VC-dimension of G_n is at most the VC-dimension of the class of all ORs of two halfspaces. Applying Lemma 26 completes the proof. ∎

Applying Lemmas 25 and 27, we obtain Lemma 7.

Appendix B. Proof of Lemma 8

We will use the following, which strengthens bounds like Lemma 25 when the expectations being estimated are small. It differs from most bounds of this type by providing an especially strong bound on the probability that the estimates are *much* larger than the true expectations.

Lemma 28 (Bshouty et al. 2009) *Suppose F is a set of $\{0,1\}$-valued functions with a common domain X. Let d be the VC-dimension of F. Let $\mathcal{D}$ be a probability distribution over X. Choose $\alpha > 0$ and $K \geq 4$. Then if*

$$m \geq \frac{c\left(d\log\frac{1}{\alpha} + \log\frac{1}{\delta}\right)}{\alpha K \log K},$$

where c is an absolute constant, then

$$\Pr_{\mathbf{u}\sim\mathcal{D}^m}[\exists f \in F,\ \mathbf{E}_{\mathcal{D}}(f) \leq \alpha \textit{ but } \hat{\mathbf{E}}_{\mathbf{u}}(f) > K\alpha] \leq \delta,$$

where $\hat{\mathbf{E}}_{\mathbf{u}}(f) = \frac{1}{m}\sum_{i=1}^{m} f(u_i)$.

To prove Lemma 8, we first use the fact that, for any fixed $\mathbf{a} \in \mathbb{S}^{n-1}$ and $\beta > 0$, it is known (see Kalai et al. 2008) that

$$\Pr_{x\in\mathbb{S}^{n-1}}[|\mathbf{a}\cdot\mathbf{x}| > \beta] \leq e^{-\beta^2 n/2}.$$

Further, as in the proof of Lemma 7, we have that

$$|\mathbf{a}\cdot\mathbf{x}| > \beta \quad \text{if and only if} \quad \mathbf{a}\cdot\mathbf{x} > \beta \text{ OR } (-\mathbf{a})\cdot\mathbf{x} > \beta,$$

so that the set of events whose probabilities we need to estimate is contained in the set of unions of pairs of halfspaces. Applying Lemma 26 and Lemma 28 completes the proof.

References

S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 724–733, 1993.

A. Blum, A. Frieze, R. Kannan, and S. Vempala. A polynomial time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1/2):35–52, 1997.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

S. C. Brubaker. *Extensions of Principle Components Analysis*. PhD thesis, Georgia Institute of Technology, 2009.

N. H. Bshouty, Y. Li, and P. M. Long. Using the doubling dimension to analyze the generalization of learning algorithms. *Journal of Computer & System Sciences*, 75(6):323–335, 2009.

J. Dunagan and S. Vempala. Optimal outlier removal in high-dimensional spaces. *J. Computer & System Sciences*, 68(2):335–373, 2004.

V. Feldman, P. Gopalan, S. Khot, and A. Ponnuswami. New results for learning noisy parities and halfspaces. In *Proc. 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 563–576, 2006.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

D. Gavinsky. Optimally-smooth adaptive boosting and application to agnostic learning. *Journal of Machine Learning Research*, 4:101–117, 2003.

V. Guruswami and P. Raghavendra. Hardness of learning halfspaces with noise. In *Proc. 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 543–552. IEEE Computer Society, 2006.

D. Haussler, M. Kearns, N. Littlestone, and M. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161, 1991.

I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics, 2002.

A. Kalai, A. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008.

M. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.

M. Kearns, R. Schapire, and L. Sellie. Toward Efficient Agnostic Learning. *Machine Learning*, 17 (2/3):115–141, 1994.

A. Klivans and A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. In *Proc. 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 553–562, 2006.

N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1987.

N. Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using Winnow. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 147–156, 1991.

L. Lovász and S. Vempala. The geometry of logconcave functions and sampling algorithms. *Random Structures and Algorithms*, 30(3):307–358, 2007.

W. Maass and G. Turan. How fast can a threshold gate learn? In *Computational Learning Theory and Natural Learning Systems: Volume I: Constraints and Prospects*, pages 381–414. MIT Press, 1994.

Y. Mansour and M. Parnas. Learning conjunctions with noise under product distributions. *Information Processing Letters*, 68(4):189–196, 1998.

A. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on Mathematical Theory of Automata*, volume XII, pages 615–622, 1962.

D. Pollard. *Convergence of Stochastic Processes*. Springer Verlag, 1984.

F. Rosenblatt. The Perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

R. Servedio. *Efficient Algorithms in Computational Learning Theory*. PhD thesis, Harvard University, 2001.

R. Servedio. PAC analogues of Perceptron and Winnow via boosting the margin. *Machine Learning*, 47(2/3):133–151, 2002.

R. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.

J. Shawe-Taylor and N. Cristianini. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

M. Talagrand. Sharper bounds for Gaussian and empirical processes. *Annals of Probability*, 22: 28–76, 1994.

L. Valiant. Learning disjunctions of conjunctions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 560–566, 1985.

H. Xu, C. Caramanis, and S. Mannor. Principal component analysis with contaminated data: The high dimensional case. *Journal of Machine Learning Research*, 2009. To appear.

Journal of Machine Learning Research 10 (2009) 2741-2775 Submitted 2/09; Revised 7/09; Published 12/09

Reproducing Kernel Banach Spaces for Machine Learning

Haizhang Zhang HZHANG12@SYR.EDU
Yuesheng Xu YXU06@SYR.EDU
Department of Mathematics
Syracuse University
Syracuse, NY 13244, USA

Jun Zhang JUNZ@UMICH.EDU
Department of Psychology
University of Michigan
Ann Arbor, MI 48109, USA

Editor: Ingo Steinwart

Abstract

We introduce the notion of reproducing kernel Banach spaces (RKBS) and study special semi-inner-product RKBS by making use of semi-inner-products and the duality mapping. Properties of an RKBS and its reproducing kernel are investigated. As applications, we develop in the framework of RKBS standard learning schemes including minimal norm interpolation, regularization network, support vector machines, and kernel principal component analysis. In particular, existence, uniqueness and representer theorems are established.

Keywords: reproducing kernel Banach spaces, reproducing kernels, learning theory, semi-inner-products, representer theorems

1. Introduction

Learning a function from its finite samples is a fundamental science problem. The essence in achieving this is to choose an appropriate measurement of similarities between elements in the domain of the function. A recent trend in machine learning is to use a positive definite kernel (Aronszajn, 1950) to measure the similarity between elements in an input space X (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004; Vapnik, 1998; Xu and Zhang, 2007, 2009). Set $\mathbb{N}_n := \{1,2,\ldots,n\}$ for $n \in \mathbb{N}$. A function $K : X \times X \to \mathbb{C}$ is called a *positive definite kernel* if for all finite subsets $\mathbf{x} := \{x_j : j \in \mathbb{N}_n\} \subseteq X$ the matrix

$$K[\mathbf{x}] := [K(x_j, x_k) : j,k \in \mathbb{N}_n] \tag{1}$$

is hermitian and positive semi-definite. The reason of using positive definite kernels to measure similarity lies in the celebrated theoretical fact due to Mercer (1909) that there is a bijective correspondence between them and *reproducing kernel Hilbert spaces* (RKHS). An RKHS $\mathcal{H}$ on X is a Hilbert space of functions on X for which point evaluations are always continuous linear functionals. One direction of the bijective correspondence says that if K is a positive definite kernel on X then there exists a unique RKHS $\mathcal{H}$ on X such that $K(x,\cdot) \in \mathcal{H}$ for each $x \in X$ and for all $f \in \mathcal{H}$ and $y \in X$

$$f(y) = (f, K(y,\cdot))_{\mathcal{H}}, \tag{2}$$

where $(\cdot,\cdot)_{\mathcal{H}}$ denotes the inner product on $\mathcal{H}$. Conversely, if $\mathcal{H}$ is an RKHS on X then there is a unique positive definite kernel K on X such that $\{K(x,\cdot) : x \in X\} \subseteq \mathcal{H}$ and (2) holds. In light of this bijective correspondence, positive definite kernels are usually called *reproducing kernels*.

By taking $f := K(x,\cdot)$ for $x \in X$ in Equation (2), we get that

$$K(x,y) = (K(x,\cdot),K(y,\cdot))_{\mathcal{H}}, \ \ x,y \in X. \tag{3}$$

Thus $K(x,y)$ is represented as an inner product on an RKHS. This explains why $K(x,y)$ is able to measure similarities of x and y. The advantages brought by the use of an RKHS include: (1) the inputs can be handled and explained geometrically; (2) geometric objects such as hyperplanes are provided by the RKHS for learning; (3) the powerful tool of functional analysis applies (Schölkopf and Smola, 2002). Based on the theory of reproducing kernels, many effective schemes have been developed for learning from finite samples (Evgeniou et al., 2000; Micchelli et al., 2009; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004; Vapnik, 1998). In particular, the widely used regularized learning algorithm works by generating a predictor function from the training data $\{(x_j,y_j) : j \in \mathbb{N}_n\} \subseteq X \times \mathbb{C}$ as the minimizer of

$$\min_{f \in \mathcal{H}_K} \sum_{j \in \mathbb{N}_n} \mathcal{L}(f(x_j),y_j) + \mu \|f\|^2_{\mathcal{H}_K}, \tag{4}$$

where $\mathcal{H}_K$ denotes the RKHS corresponding to the positive definite kernel K, $\mathcal{L}$ is a prescribed loss function, and μ is a positive regularization parameter.

This paper is motivated from machine learning in Banach spaces. There are advantages of learning in Banach spaces over Hilbert spaces. Firstly, there is essentially only one Hilbert space once the dimension of the space is fixed. This follows from the well-known fact that any two Hilbert spaces over $\mathbb{C}$ of the same dimension are isometrically isomorphic. By contrast, for $p \neq q \in [1,+\infty]$, $L^p[0,1]$ and $L^q[0,1]$ are not isomorphic, namely, there does not exist a bijective bounded linear mapping between them (see, Fabian et al., 2001, page 180). Thus, compared to Hilbert spaces, Banach spaces possess much richer geometric structures, which are potentially useful for developing learning algorithms. Secondly, in some applications, a norm from a Banach space is invoked without being induced from an inner product. For instance, it is known that minimizing about the ℓ^p norm on $\mathbb{R}^d$ leads to sparsity of the minimizer when p is close to 1 (see, for example, Tropp, 2006). In the extreme case that $\varphi : \mathbb{R}^d \to [0,+\infty)$ is strictly concave and $\mu > 0$, one can show that the minimizer for

$$\min\{\varphi(x) + \mu\|x\|_{\ell^1} : x \in \mathbb{R}^d\} \tag{5}$$

has at most one nonzero element. The reason is that the extreme points on a sphere in the ℓ^1 norm must lie on axes of the Euclidean coordinate system. A detailed proof of this result is provided in the appendix. Thirdly, since many training data come with intrinsic structures that make them impossible to be embedded into a Hilbert space, learning algorithms based on RKHS may not work well for them. Hence, there is a need to modify the algorithms by adopting norms in Banach spaces. For example, one might have to replace the norm $\|\cdot\|_{\mathcal{H}_K}$ in (4) with that of a Banach space.

There has been considerable work on learning in Banach spaces in the literature. References Bennett and Bredensteiner (2000); Micchelli and Pontil (2004, 2007); Micchelli et al. (2003); Zhang (2002) considered the problem of minimizing a regularized functional of the form

$$\sum_{j \in \mathbb{N}_n} \mathcal{L}(\lambda_j(f),y_j) + \phi(\|f\|_{\mathcal{B}}), \ \ f \in \mathcal{B},$$

where $\mathcal{B}$ is Banach space, λ_j are in the dual $\mathcal{B}^*$, $y_j \in \mathbb{C}$, $\mathcal{L}$ is a loss function, and ϕ is a strictly increasing nonnegative function. In particular, Micchelli et al. (2003) considered learning in Besov spaces (a special type of Banach spaces). On-line learning in finite dimensional Banach spaces was studied, for example, in Gentile (2001). Learning of an L^p function was considered in Kimber and Long (1995). Classifications in Banach spaces, and more generally in metric spaces were discussed in Bennett and Bredensteiner (2000), Der and Lee (2007), Hein et al. (2005), von Luxburg and Bousquet (2004) and Zhou et al. (2002).

The above discussion indicates that there is a need of introducing the notion of reproducing kernel Banach spaces for the systematic study of learning in Banach spaces. Such a definition is expected to result in consequences similar to those in an RKHS. A generalization of RKHS to non-Hilbert spaces using point evaluation with kernels was proposed in Canu et al. (2003), although the spaces considered there might be too general to have favorable properties of an RKHS. We shall introduce the notion of reproducing kernel Banach spaces in Section 2, and a general construction in Section 3. It will become clear that the lack of an inner product may cause arbitrariness in the properties of the associated reproducing kernel. To overcome this, we shall establish in Section 4 s.i.p. reproducing kernel Banach spaces by making use of semi-inner-products for normed vector spaces first defined by Lumer (1961) and further developed by Giles (1967). Semi-inner-products were first applied to machine learning by Der and Lee (2007) to develop hard margin hyperplane classification in Banach spaces. Here the availability of a semi-inner-product enables us to study basic properties of reproducing kernel Banach spaces and their reproducing kernels. In Section 5, we shall develop in the framework of reproducing kernel Banach spaces standard learning schemes including minimal norm interpolation, regularization network, support vector machines, and kernel principal component analysis. Existence, uniqueness and representer theorems for the learning schemes will be proved. We draw conclusive remarks in Section 6 and include two technical results in Appendix.

2. Reproducing Kernel Banach Spaces

Without specifically mentioned, all vector spaces in this paper are assumed to be complex. Let X be a prescribed input space. A normed vector space $\mathcal{B}$ is called a **Banach space of functions** on X if it is a Banach space whose elements are functions on X, and for each $f \in \mathcal{B}$, its norm $\|f\|_{\mathcal{B}}$ in $\mathcal{B}$ vanishes if and only if f, as a function, vanishes everywhere on X. By this definition, $L^p[0,1]$, $1 \le p \le +\infty$, is not a Banach space of functions as it consists of equivalent classes of functions with respect to the Lebesgue measure.

Influenced by the definition of RKHS, our first intuition is to define a reproducing kernel Banach space (RKBS) as a Banach space of functions on X on which point evaluations are continuous linear functionals. If such a definition was adopted then the first example that comes to our mind would be $C[0,1]$, the Banach space of continuous functions on $[0,1]$ equipped with the maximum norm. It satisfies the definition. However, since for each $f \in C[0,1]$,

$$f(x) = \delta_x(f), \quad x \in [0,1],$$

the reproducing kernel for $C[0,1]$ would have to be the delta distribution, which is not a function that can be evaluated. This example suggests that there should exist a way of identifying the elements in the dual of an RKBS with functions. Recall that two normed vector spaces V_1 and V_2 are said to be *isometric* if there is a bijective linear norm-preserving mapping between them. We call such

V_1 and V_2 an *identification* of each other. We would like the dual space $\mathcal{B}^*$ of an RKBS $\mathcal{B}$ on X to be isometric to a Banach space of functions on X. In addition to this requirement, later on we will find it very convenient to jump freely between a Banach space and its dual. For this reason, we would like an RKBS $\mathcal{B}$ to be *reflexive* in the sense that $(\mathcal{B}^*)^* = \mathcal{B}$. The above discussion leads to the following formal definition.

Definition 1 *A* **reproducing kernel Banach space** *(RKBS) on X is a reflexive Banach space $\mathcal{B}$ of functions on X for which $\mathcal{B}^*$ is isometric to a Banach space $\mathcal{B}^\#$ of functions on X and the point evaluation is continuous on both $\mathcal{B}$ and $\mathcal{B}^\#$.*

Several remarks are in order about this definition. First, whether $\mathcal{B}$ is an RKBS is independent of the choice of the identification $\mathcal{B}^\#$ of $\mathcal{B}^*$. In other words, if the point evaluation is continuous on some identification $\mathcal{B}^\#$ then it is continuous on all the identifications. The reason is that any two identifications of $\mathcal{B}^*$ are isometric. Second, an RKHS $\mathcal{H}$ on X is an RKBS. To see this, we set

$$\mathcal{H}^\# := \{\bar{f} : f \in \mathcal{H}\} \tag{6}$$

with the norm $\|\bar{f}\|_{\mathcal{H}^\#} := \|f\|_{\mathcal{H}}$, where $\bar{f}$ denotes the conjugate of f defined by $\bar{f}(x) := \overline{f(x)}, x \in X$. By the Riesz representation theorem (Conway, 1990), each $u^* \in \mathcal{H}^*$ has the form

$$u^*(f) = (f, f_0)_{\mathcal{H}}, \ \ f \in \mathcal{H}$$

for some unique $f_0 \in \mathcal{H}$ and $\|u^*\|_{\mathcal{H}^*} = \|f_0\|_{\mathcal{H}}$. We introduce a mapping $\iota : \mathcal{H}^* \to \mathcal{H}^\#$ by setting

$$\iota(u^*) := \overline{f_0}.$$

Clearly, ι so defined is isometric from $\mathcal{H}^*$ to $\mathcal{H}^\#$. We conclude that an RKHS is a special RKBS. Third, the identification $\mathcal{B}^\#$ of $\mathcal{B}^*$ of an RKBS is usually not unique. However, since they are isometric to each other, we shall assume that one of them has been chosen for an RKBS $\mathcal{B}$ under discussion. In particular, the identification of $\mathcal{H}^*$ of an RKHS $\mathcal{H}$ will always be chosen as (6). Fourth, for notational simplicity, we shall still denote the fixed identification of $\mathcal{B}^*$ by $\mathcal{B}^*$. Let us keep in mind that originally $\mathcal{B}^*$ consists of continuous linear functionals on $\mathcal{B}$. Thus, when we shall be treating elements in $\mathcal{B}^*$ as functions on X, we actually think $\mathcal{B}^*$ as its chosen identification. With this notational convention, we state our last remark that if $\mathcal{B}$ is an RKBS on X then so is $\mathcal{B}^*$.

We shall show that there indeed exists a *reproducing kernel* for an RKBS. To this end, we introduce for a normed vector space V the following *bilinear form* on $V \times V^*$ by setting

$$(u, v^*)_V := v^*(u), \ \ u \in V, \ v^* \in V^*.$$

It is called bilinear for the reason that for all $\alpha, \beta \in \mathbb{C}$, $u, v \in V$, and $u^*, v^* \in V^*$ there holds

$$(\alpha u + \beta v, u^*)_V = \alpha(u, u^*)_V + \beta(v, u^*)_V$$

and

$$(u, \alpha u^* + \beta v^*)_V = \alpha(u, u^*)_V + \beta(u, v^*)_V.$$

Note that if V is a reflexive Banach space then for any continuous linear functional T on V^* there exists a unique $u \in V$ such that

$$T(v^*) = (u, v^*)_V, \ \ v^* \in V^*.$$

Theorem 2 *Suppose that $\mathcal{B}$ is an RKBS on X. Then there exists a unique function $K : X \times X \to \mathbb{C}$ such that the following statements hold.*
(a) For every $x \in X$, $K(\cdot,x) \in \mathcal{B}^*$ and

$$f(x) = (f, K(\cdot,x))_{\mathcal{B}}, \ \text{for all } f \in \mathcal{B}.$$

(b) For every $x \in X$, $K(x,\cdot) \in \mathcal{B}$ and

$$f^*(x) = (K(x,\cdot), f^*)_{\mathcal{B}}, \ \text{for all } f^* \in \mathcal{B}^*. \tag{7}$$

(c) The linear span of $\{K(x,\cdot) : x \in X\}$ is dense in $\mathcal{B}$, namely,

$$\overline{\text{span}}\{K(x,\cdot) : x \in X\} = \mathcal{B}. \tag{8}$$

(d) The linear span of $\{K(\cdot,x) : x \in X\}$ is dense in $\mathcal{B}^*$, namely,

$$\overline{\text{span}}\{K(\cdot,x) : x \in X\} = \mathcal{B}^*. \tag{9}$$

(e) For all $x, y \in X$

$$K(x,y) = (K(x,\cdot), K(\cdot,y))_{\mathcal{B}}. \tag{10}$$

Proof For every $x \in X$, since δ_x is a continuous linear functional on $\mathcal{B}$, there exists $g_x \in \mathcal{B}^*$ such that

$$f(x) = (f, g_x)_{\mathcal{B}}, \ \ f \in \mathcal{B}.$$

We introduce a function $\tilde{K}$ on $X \times X$ by setting

$$\tilde{K}(x,y) := g_x(y), \ \ x, y \in X.$$

It follows that $\tilde{K}(x,\cdot) \in \mathcal{B}^*$ for each $x \in X$, and

$$f(x) = (f, \tilde{K}(x,\cdot))_{\mathcal{B}}, \ \ f \in \mathcal{B}, \ \ x \in X. \tag{11}$$

There is only one function on $X \times X$ with the above properties. Assume to the contrary that there is another $\tilde{G} : X \times X \to \mathbb{C}$ satisfying $\{\tilde{G}(x,\cdot) : x \in X\} \subseteq \mathcal{B}^*$ and

$$f(x) = (f, \tilde{G}(x,\cdot))_{\mathcal{B}}, \ \ f \in \mathcal{B}, \ \ x \in X.$$

The above equation combined with (11) yields that

$$(f, \tilde{K}(x,\cdot) - \tilde{G}(x,\cdot))_{\mathcal{B}} = 0, \ \ \text{for all } f \in \mathcal{B}, \ x \in X.$$

Thus, $\tilde{K}(x,\cdot) - \tilde{G}(x,\cdot) = 0$ in $\mathcal{B}^*$ for each $x \in X$. Since $\mathcal{B}^*$ is a Banach space of functions on X, we get for every $y \in X$ that

$$\tilde{K}(x,y) - \tilde{G}(x,y) = 0,$$

that is, $\tilde{K} = \tilde{G}$.

Likewise, there exists a unique $K : X \times X \to \mathbb{C}$ such that $K(y,\cdot) \in \mathcal{B}$, $y \in X$ and

$$f^*(y) = (K(y,\cdot), f^*)_{\mathcal{B}}, \ \ f^* \in \mathcal{B}^*, \ \ y \in X. \tag{12}$$

Letting $f := K(y,\cdot)$ in (11) yields that

$$K(y,x) = (K(y,\cdot), \tilde{K}(x,\cdot))_{\mathcal{B}}, \ \ x,y \in X, \tag{13}$$

and setting $f^* := \tilde{K}(x,\cdot)$ in (12) ensures that

$$\tilde{K}(x,y) = (K(y,\cdot), \tilde{K}(x,\cdot))_{\mathcal{B}}, \ \ x,y \in X.$$

Combining the above equation with (13), we get that

$$\tilde{K}(x,y) = K(y,x), \ \ x,y \in X.$$

Therefore, K satisfies (a) and (b) as stated in the theorem. Equation (10) in (e) is proved by letting $f^* = K(\cdot,y)$ in (7). To complete the proof, we shall show (c) only, since (d) can be handled in a similar way. Suppose that (8) does not hold. Then by the Hahn-Banach theorem, there exists a nontrivial functional $f^* \in \mathcal{B}^*$ such that

$$(K(x,\cdot), f^*)_{\mathcal{B}} = 0, \ \ \text{for all } x \in X.$$

We get immediately from (12) that $f^*(x) = 0$ for all $x \in X$. Since $\mathcal{B}^*$ is a Banach space of functions on X, $f^* = 0$ in $\mathcal{B}^*$, a contradiction. ■

We call the function K in Theorem 2 the **reproducing kernel** for the RKBS $\mathcal{B}$. By Theorem 2, an RKBS has exactly one reproducing kernel. However, different RKBS may have the same reproducing kernel. Examples will be given in the next section. This results from a fundamental difference between Banach spaces and Hilbert spaces. To explain this, we let $\mathcal{W}$ be a Banach space and V a subset of $\mathcal{W}$ such that $\operatorname{span} V$ is dense in $\mathcal{W}$. Suppose that a norm on elements of V is prescribed. If $\mathcal{W}$ is a Hilbert space and an inner product is defined among elements in V, then the norm extends in a unique way to $\operatorname{span} V$, and hence to the whole space $\mathcal{W}$. Assume now that $\mathcal{W}$ is only known to be a Banach space and $V^* \subseteq \mathcal{W}^*$ satisfying $\overline{\operatorname{span}} V^* = \mathcal{W}^*$ is given. Then even if a bilinear form is defined between elements in V and those in V^*, the norm may not have a unique extension to the whole space $\mathcal{W}$. Consequently, although we have at hand a reproducing kernel K for an RKBS $\mathcal{B}$, the relationship (13), and denseness conditions (8), (9), we still can not determine the norm on $\mathcal{B}$.

3. Construction of Reproducing Kernels via Feature Maps

In this section, we shall characterize reproducing kernels for RKBS. The characterization will at the same time provide a convenient way of constructing reproducing kernels and their corresponding RKBS. For the corresponding results in the RKHS case, see, for example, Saitoh (1997), Schölkopf and Smola (2002), Shawe-Taylor and Cristianini (2004) and Vapnik (1998).

Theorem 3 *Let $\mathcal{W}$ be a reflexive Banach space with dual space $\mathcal{W}^*$. Suppose that there exists $\Phi : X \to \mathcal{W}$, and $\Phi^* : X \to \mathcal{W}^*$ such that*

$$\overline{\operatorname{span}}\, \Phi(X) = \mathcal{W}, \ \ \overline{\operatorname{span}}\, \Phi^*(X) = \mathcal{W}^*. \tag{14}$$

Then $\mathcal{B} := \{(u, \Phi^*(\cdot))_{\mathcal{W}} : u \in \mathcal{W}\}$ *with norm*

$$\|(u, \Phi^*(\cdot))_{\mathcal{W}}\|_{\mathcal{B}} := \|u\|_{\mathcal{W}} \tag{15}$$

is an RKBS on X *with the dual space* $\mathcal{B}^* := \{(\Phi(\cdot), u^*)_{\mathcal{W}} : u^* \in \mathcal{W}^*\}$ *endowed with the norm*

$$\|(\Phi(\cdot), u^*)_{\mathcal{W}}\|_{\mathcal{B}^*} := \|u^*\|_{\mathcal{W}^*}$$

and the bilinear form

$$((u, \Phi^*(\cdot))_{\mathcal{W}}, (\Phi(\cdot), u^*)_{\mathcal{W}})_{\mathcal{B}} := (u, u^*)_{\mathcal{W}}, \ \ u \in \mathcal{W}, \ u^* \in \mathcal{W}^*. \tag{16}$$

Moreover, the reproducing kernel K *for* $\mathcal{B}$ *is*

$$K(x, y) := (\Phi(x), \Phi^*(y))_{\mathcal{W}}, \ \ x, y \in X. \tag{17}$$

Proof We first show that $\mathcal{B}$ defined above is a Banach space of functions on X. To this end, we set $u \in \mathcal{W}$ and assume that

$$(u, \Phi^*(x))_{\mathcal{W}} = 0, \ \ \text{for all } x \in X. \tag{18}$$

Then by the denseness condition (14), $(u, u^*)_{\mathcal{W}} = 0$ for all $u^* \in \mathcal{W}^*$, implying that $u = 0$. Conversely, if $u = 0$ in $\mathcal{W}$ then it is clear that (18) holds true. These arguments also show that the representer $u \in \mathcal{W}$ for a function $(u, \Phi^*(\cdot))_{\mathcal{W}}$ in $\mathcal{B}$ is unique. It is obvious that (15) defines a norm on $\mathcal{B}$ and $\mathcal{B}$ is complete under this norm. Therefore, $\mathcal{B}$ is a Banach space of functions on X. Similarly, so is $\tilde{\mathcal{B}} := \{(\Phi(\cdot), u^*)_{\mathcal{W}} : u^* \in \mathcal{W}^*\}$ equipped with the norm

$$\|(\Phi(\cdot), u^*)_{\mathcal{W}}\|_{\tilde{\mathcal{B}}} := \|u^*\|_{\mathcal{W}^*}.$$

Define the bilinear form T on $\mathcal{B} \times \tilde{\mathcal{B}}$ by setting

$$T((u, \Phi^*(\cdot))_{\mathcal{W}}, (\Phi(\cdot), u^*)_{\mathcal{W}}) := (u, u^*)_{\mathcal{W}}, \ \ u \in \mathcal{W}, \ u^* \in \mathcal{W}^*.$$

Clearly, we have for all $u \in \mathcal{W}$, $u^* \in \mathcal{W}^*$ that

$$|T((u, \Phi^*(\cdot))_{\mathcal{W}}, (\Phi(\cdot), u^*)_{\mathcal{W}})| \le \|u\|_{\mathcal{W}} \|u^*\|_{\mathcal{W}^*} = \|(u, \Phi^*(\cdot))_{\mathcal{W}}\|_{\mathcal{B}} \, \|(\Phi(\cdot), u^*)_{\mathcal{W}}\|_{\tilde{\mathcal{B}}}.$$

Therefore, each function in $\tilde{\mathcal{B}}$ is a continuous linear functional on $\mathcal{B}$. Note that the linear mapping $u \to (u, \Phi^*(\cdot))_{\mathcal{W}}$ is isometric from $\mathcal{W}$ to $\mathcal{B}$. As a consequence, functions in $\tilde{\mathcal{B}}$ exhaust all the continuous linear functionals on $\mathcal{B}$. We conclude that $\mathcal{B}^* = \tilde{\mathcal{B}}$ with the bilinear form (16). Likewise, one can show that $\mathcal{B}$ is the dual of $\mathcal{B}^*$ by the reflexivity of $\mathcal{W}$. We have hence proved that $\mathcal{B}$ is reflexive with dual $\mathcal{B}^*$.

It remains to show that point evaluations are continuous on $\mathcal{B}$ and $\mathcal{B}^*$. To this end, we get for each $x \in X$ and $f := (u, \Phi^*(\cdot))_{\mathcal{W}}, u \in \mathcal{W}$ that

$$|f(x)| = |(u, \Phi^*(x))_{\mathcal{W}}| \le \|u\|_{\mathcal{W}} \|\Phi^*(x)\|_{\mathcal{W}^*} = \|f\|_{\mathcal{B}} \|\Phi^*(x)\|_{\mathcal{W}^*},$$

which implies that δ_x is continuous on $\mathcal{B}$. By similar arguments, it is continuous on $\mathcal{B}^*$. Combining all the discussion above, we reach the conclusion that $\mathcal{B}$ is an RKBS on X.

For the function K on $X \times X$ defined by (17), we get that $K(x,\cdot) \in \mathcal{B}$ and $K(\cdot,x) \in \mathcal{B}^*$ for all $x \in X$. It is also verified that for $f := (u, \Phi^*(\cdot))_{\mathcal{W}}$, $u \in \mathcal{W}$

$$(f, K(\cdot,x))_{\mathcal{B}} = ((u,\Phi^*(\cdot))_{\mathcal{W}}, (\Phi(\cdot),\Phi^*(x))_{\mathcal{W}})_{\mathcal{B}} = (u,\Phi^*(x))_{\mathcal{W}} = f(x).$$

Similarly, for $f^* := (\Phi(\cdot), u^*)_{\mathcal{W}}$, $u^* \in \mathcal{W}^*$

$$(K(x,\cdot), f^*)_{\mathcal{B}} = ((\Phi(x),\Phi^*(\cdot))_{\mathcal{W}}, (\Phi(\cdot),u^*)_{\mathcal{W}})_{\mathcal{B}} = (\Phi(x),u^*)_{\mathcal{W}} = f^*(x).$$

These facts show that K is the reproducing kernel for $\mathcal{B}$ and complete the proof. ■

We call the mappings Φ, Φ^* in Theorem 3 a pair of **feature maps** for the reproducing kernel K. The spaces $\mathcal{W}$, $\mathcal{W}^*$ are called the pair of **feature spaces** associated with the feature maps for K. As a corollary to Theorem 3, we obtain the following characterization of reproducing kernels for RKBS.

Theorem 4 *A function $K : X \times X \to \mathbb{C}$ is the reproducing kernel of an RKBS on X if and only if it is of the form (17), where $\mathcal{W}$ is a reflexive Banach space, and mappings $\Phi : X \to \mathcal{W}$, $\Phi^* : X \to \mathcal{W}^*$ satisfy (14).*

Proof The sufficiency has been shown by the last theorem. For the necessity, we assume that K is the reproducing kernel of an RKBS $\mathcal{B}$ on X, and set

$$\mathcal{W} := \mathcal{B}, \;\; \mathcal{W}^* := \mathcal{B}^*, \;\; \Phi(x) := K(x,\cdot), \;\; \Phi^*(x) := K(\cdot,x), \;\; x \in X.$$

By Theorem 2, $\mathcal{W}, \mathcal{W}^*, \Phi, \Phi^*$ satisfy all the conditions. ■

To demonstrate how we get RKBS and their reproducing kernels by Theorem 3, we now present a nontrivial example of RKBS. Set $X := \mathbb{R}, \mathbb{I} := [-\frac{1}{2}, \frac{1}{2}]$, and $p \in (1, +\infty)$. We make the convention that q is always the conjugate number of p, that is, $p^{-1} + q^{-1} = 1$. Define $\mathcal{W} := L^p(\mathbb{I})$, $\mathcal{W}^* := L^q(\mathbb{I})$ and $\Phi : X \to \mathcal{W}$, $\Phi^* : X \to \mathcal{W}^*$ as

$$\Phi(x)(t) := e^{-i2\pi xt}, \; \Phi^*(x)(t) := e^{i2\pi xt}, \quad x \in \mathbb{R}, \, t \in \mathbb{I}.$$

For $f \in L^1(\mathbb{R})$, its Fourier transform $\hat{f}$ is defined as

$$\hat{f}(t) := \int_{\mathbb{R}} f(x) e^{-i2\pi xt} dx, \;\; t \in \mathbb{R},$$

and its inverse Fourier transform $\check{f}$ is defined by

$$\check{f}(t) := \int_{\mathbb{R}} f(x) e^{i2\pi xt} dx, \;\; t \in \mathbb{R}.$$

The Fourier transform and the inverse Fourier transform can be defined on tempered distributions. Since the Fourier transform is injective on $L^1(\mathbb{R})$ (see, Rudin, 1987, page 185), the denseness requirement (14) is satisfied.

By the construction described in Theorem 3, we obtain

$$\mathcal{B} := \{f \in C(\mathbb{R}) : \operatorname{supp} \hat{f} \subseteq \mathbb{I},\ \hat{f} \in L^p(\mathbb{I})\} \tag{19}$$

with norm $\|f\|_{\mathcal{B}} := \|\hat{f}\|_{L^p(\mathbb{I})}$, and the dual

$$\mathcal{B}^* := \{g \in C(\mathbb{R}) : \operatorname{supp} \hat{g} \subseteq \mathbb{I},\ \hat{g} \in L^q(\mathbb{I})\}$$

with norm $\|g\|_{\mathcal{B}^*} := \|\check{g}\|_{L^q(\mathbb{I})}$. For each $f \in \mathcal{B}$ and $g \in \mathcal{B}^*$, we have

$$(f,g)_{\mathcal{B}} = \int_{\mathbb{I}} \hat{f}(t)\check{g}(t)dt.$$

The kernel K for $\mathcal{B}$ is given as

$$K(x,y) = (\Phi(x), \Phi^*(y))_{\mathcal{W}} = \int_{\mathbb{I}} e^{-i2\pi xt} e^{i2\pi yt} dt = \frac{\sin \pi(x-y)}{\pi(x-y)} = \operatorname{sinc}(x-y).$$

We check that for each $f \in \mathcal{B}$

$$(f, K(\cdot,x))_{\mathcal{B}} = \int_{\mathbb{I}} \hat{f}(t)(K(\cdot,x))\check{}(t)dt = \int_{\mathbb{I}} \hat{f}(t)e^{i2\pi xt}dt = f(x),\ \ x \in \mathbb{R}$$

and for each $g \in \mathcal{B}^*$

$$(K(x,\cdot), g)_{\mathcal{B}} = \int_{\mathbb{I}} (K(x,\cdot))\hat{}(t)\check{g}(t)dt = \int_{\mathbb{I}} \check{g}(t)e^{-i2\pi xt}dt = g(x),\ \ x \in \mathbb{R}.$$

When $p = q = 2$, $\mathcal{B}$ reduces to the classical space of bandlimited functions.

In the above example, $\mathcal{B}$ is isometrically isomorphic to $L^p(\mathbb{I})$. As mentioned in the introduction, $L^p(\mathbb{I})$ with different p are not isomorphic to each other. As a result, for different indices p the spaces $\mathcal{B}$ defined by (19) are essentially different. However, we see that they all have the sinc function as the reproducing kernel. In fact, if no further conditions are imposed on an RKBS, its reproducing kernel can be rather arbitrary. We make a simple observation below to illustrate this.

Proposition 5 *If the input space X is a finite set, then any nontrivial function K on $X \times X$ is the reproducing kernel of some RKBS on X.*

Proof Let K be an arbitrary nontrivial function on $X \times X$. Assume that $X = \mathbb{N}_m$ for some $m \in \mathbb{N}$. Let $d \in \mathbb{N}$ be the rank of the matrix $K[X]$ as defined by (1). By elementary linear algebra, there exist nonsingular matrices $P, Q \in \mathbb{C}^{m\times m}$ such that the transpose $(K[X])^T$ of $K[X]$ has the form

$$(K[X])^T = P \begin{bmatrix} I_d & 0 \\ 0 & 0 \end{bmatrix} Q = P \begin{bmatrix} I_d \\ 0 \end{bmatrix} \begin{bmatrix} I_d & 0 \end{bmatrix} Q, \tag{20}$$

where I_d is the $d \times d$ identity matrix. For $j \in \mathbb{N}_m$, let P_j be the transpose of the jth row of $P \begin{bmatrix} I_d \\ 0 \end{bmatrix}$ and Q_j the jth column of $\begin{bmatrix} I_d & 0 \end{bmatrix} Q$. Choose an arbitrary $p \in (1, +\infty)$. Equation (20) is rewritten as

$$K(j,k) = (Q_j, P_k)_{l^p(\mathbb{N}_d)},\ \ j,k \in \mathbb{N}_m. \tag{21}$$

We set $\mathcal{W} := l^p(\mathbb{N}_d)$, $\mathcal{W}^* := l^q(\mathbb{N}_d)$ and $\Phi(j) := Q_j$, $\Phi^*(j) := P_j$, $j \in \mathbb{N}_m$. Since P, Q are nonsingular, (14) holds true. Also, we have by (21) that

$$K(j,k) = (\Phi(j), \Phi^*(k))_{\mathcal{W}}, \quad j,k \in \mathbb{N}_m.$$

By Theorem 4, K is a reproducing kernel for some RKBS on X. ■

Proposition 5 reveals that due to the lack of an inner product, the reproducing kernel for a general RKBS can be an arbitrary function on $X \times X$. Particularly, it might be nonsymmetric or non-positive definite. In order for reproducing kernels of RKBS to have desired properties as those of RKHS, we may need to impose certain structures on RKBS, which in some sense are substitutes of the inner product for RKHS. For this purpose, we shall adopt the semi-inner-product introduced by Lumer (1961). A semi-inner-product possesses some but not all properties of an inner product. Hilbert space type arguments and results become available with the presence of a semi-inner-product. We shall introduce the notion of semi-inner-product RKBS.

4. S.i.p. Reproducing Kernel Banach Spaces

The purpose of this section is to establish the notion of semi-inner-product RKBS and study its properties. We start with necessary preliminaries on semi-inner-products (Giles, 1967; Lumer, 1961).

4.1 Semi-Inner-Products

A **semi-inner-product** on a vector space V is a function, denoted by $[\cdot,\cdot]_V$, from $V \times V$ to $\mathbb{C}$ such that for all $x, y, z \in V$ and $\lambda \in \mathbb{C}$

1. $[x+y,z]_V = [x,z]_V + [y,z]_V$,
2. $[\lambda x, y]_V = \lambda [x,y]_V$, $[x, \lambda y]_V = \bar{\lambda}[x,y]_V$,
3. $[x,x]_V > 0$ for $x \neq 0$,
4. (Cauchy-Schwartz) $|[x,y]_V|^2 \leq [x,x]_V [y,y]_V$.

The property that $[x, \lambda y]_V = \bar{\lambda}[x,y]_V$ was not required in the original definition by Lumer (1961). We include it here for the observation by Giles (1967) that this property can always be imposed.

It is necessary to point out the difference between a semi-inner-product and an inner product. In general, a semi-inner-product $[\cdot,\cdot]_V$ does not satisfy the conjugate symmetry $[x,y]_V = \overline{[y,x]_V}$ for all $x, y \in V$. As a consequence, there always exist $x, y, z \in V$ such that

$$[x, y+z]_V \neq [x,y]_V + [x,z]_V.$$

In fact, a semi-inner-product is always additive about the second variable only if it degenerates to an inner product. We show this fact below.

Proposition 6 *A semi-inner-product $[\cdot,\cdot]_V$ on a complex vector space V is an inner product if and only if*

$$[x, y+z]_V = [x,y]_V + [x,z]_V, \text{ for all } x, y, z \in V. \tag{22}$$

Proof Suppose that V has a semi-inner-product $[\cdot,\cdot]_V$ that satisfies (22). It suffices to show that for all $x, y \in V$,

$$[x,y]_V = \overline{[y,x]_V}. \tag{23}$$

Set $\lambda \in \mathbb{C}$. By the linearity on the first and the additivity on the second variable, we get that

$$[x+\lambda y, x+\lambda y]_V = [x,x]_V + [\lambda y, \lambda y]_V + \lambda [y,x]_V + \bar{\lambda}[x,y]_V.$$

Since $[z,z]_V \geq 0$ for all $z \in V$, we must have

$$\lambda [y,x]_V + \bar{\lambda}[x,y]_V \in \mathbb{R}.$$

Choosing $\lambda = 1$ yields that $\operatorname{Im}[y,x]_V = -\operatorname{Im}[x,y]_V$. And the choice $\lambda = i$ results that $\operatorname{Re}[y,x]_V = \operatorname{Re}[x,y]_V$. Therefore, (23) holds, which implies that $[\cdot,\cdot]_V$ is an inner product on V. ■

It was shown in Lumer (1961) that a vector space V with a semi-inner-product is a normed space equipped with

$$\|x\|_V := [x,x]_V^{1/2}, \ \ x \in V. \tag{24}$$

Therefore, if a vector space V has a semi-inner-product, we always assume that its norm is induced by (24) and call V an **s.i.p. space**. Conversely, every normed vector space V has a semi-inner-product that induces its norm by (24) (Giles, 1967; Lumer, 1961). By the Cauchy-Schwartz inequality, if V is an s.i.p. space then for each $x \in V$, $y \to [y,x]_V$ is a continuous linear functional on V. We denote this linear functional by x^*. Following this definition, we have that

$$[x,y]_V = y^*(x) = (x, y^*)_V, \ \ x, y \in V. \tag{25}$$

In general, a semi-inner-product for a normed vector space may not be unique. However, a differentiation property of the norm will ensure the uniqueness. We call a normed vector space V **Gâteaux differentiable** if for all $x, y \in V \setminus \{0\}$

$$\lim_{t \in \mathbb{R}, t \to 0} \frac{\|x+ty\|_V - \|x\|_V}{t}$$

exists. It is called **uniformly Fréchet differentiable** if the limit is approached uniformly on $\mathcal{S}(V) \times \mathcal{S}(V)$. Here, $S(V) := \{u \in V : \|u\|_V = 1\}$ is the unit sphere of V. The following result is due to Giles (1967).

Lemma 7 *If an s.i.p. space V is Gâteaux differentiable then for all $x, y \in V$ with $x \neq 0$*

$$\lim_{t \in \mathbb{R}, t \to 0} \frac{\|x+ty\|_V - \|x\|_V}{t} = \frac{\operatorname{Re}[y,x]}{\|x\|_V}. \tag{26}$$

The above lemma indicates that a Gâteaux differentiable normed vector space has a unique semi-inner-product. In fact, we have by (26) that

$$[x,y]_V = \|y\|_V \left(\lim_{t \in \mathbb{R}, t \to 0} \frac{\|y+tx\|_V - \|y\|_V}{t} + i \lim_{t \in \mathbb{R}, t \to 0} \frac{\|iy+tx\|_V - \|y\|_V}{t} \right), \ \ x, y \in V \setminus \{0\}. \tag{27}$$

For this reason, if V is a Gâteaux differentiable normed vector space we always assume that it is an s.i.p. space with the semi-inner-product defined as above. Interested readers are referred to the appendix for a proof that (27) indeed defines an s.i.p.

We shall impose one more condition on an s.i.p. space that will lead to a Riesz representation theorem. A normed vector space V is **uniformly convex** if for all $\varepsilon > 0$ there exists a $\delta > 0$ such that

$$\|x+y\|_V \leq 2-\delta \text{ for all } x,y \in \mathcal{S}(V) \text{ with } \|x-y\|_V \geq \varepsilon.$$

The space $L^p(\Omega,\mu)$, $1 < p < +\infty$, on a measure space $(\Omega,\mathcal{F},\mu)$ is uniformly convex. In particular, by the parallelogram law, any inner product space is uniformly convex. By a remark in Conway (1990), page 134, a uniformly convex Banach space is reflexive. There is a well-known relationship between uniform Fréchet differentiability and uniform convexity (Cudia, 1963). It states that a normed vector space is uniformly Fréchet differentiable if and only if its dual is uniformly convex. Therefore, if $\mathcal{B}$ is a uniformly convex and uniformly Fréchet differentiable Banach space then so is $\mathcal{B}^*$ since $\mathcal{B}$ is reflexive. The important role of uniform convexity is displayed in the next lemma (Giles, 1967).

Lemma 8 *(Riesz Representation Theorem) Suppose that $\mathcal{B}$ is a uniformly convex, uniformly Fréchet differentiable Banach space. Then for each $f \in \mathcal{B}^*$ there exists a unique $x \in \mathcal{B}$ such that $f = x^*$, that is,*

$$f(y) = [y,x]_{\mathcal{B}}, \ \ y \in \mathcal{B}.$$

Moreover, $\|f\|_{\mathcal{B}^*} = \|x\|_{\mathcal{B}}$.

The above Riesz representation theorem is desirable for RKBS. By Lemma 8 and the discussion right before it, we shall investigate in the next subsection RKBS which are both uniformly convex and uniformly Fréchet differentiable.

Let $\mathcal{B}$ be a uniformly convex and uniformly Fréchet differentiable Banach space. By Lemma 8, $x \to x^*$ defines a bijection from $\mathcal{B}$ to $\mathcal{B}^*$ that preserves the norm. Note that this **duality mapping** is in general nonlinear. We call x^* the **dual element** of x. Since $\mathcal{B}^*$ is uniformly Fréchet differentiable, it has a unique semi-inner-product, which is given by

$$[x^*,y^*]_{\mathcal{B}^*} = [y,x]_{\mathcal{B}}, \ \ x,y \in \mathcal{B}. \tag{28}$$

We close this subsection with a concrete example of uniformly convex and uniformly Fréchet differentiable Banach spaces. Let $(\Omega,\mathcal{F},\mu)$ be a measure space and $\mathcal{B} := L^p(\Omega,\mu)$ for some $p \in (1,+\infty)$. It is uniformly convex and uniformly Fréchet differentiable with dual $\mathcal{B}^* = L^q(\Omega,\mu)$. For each $f \in \mathcal{B}$, its dual element in $\mathcal{B}^*$ is

$$f^* = \frac{\bar{f}|f|^{p-2}}{\|f\|^{p-2}_{L^p(\Omega,\mu)}}. \tag{29}$$

Consequently, the semi-inner-product on $\mathcal{B}$ is

$$[f,g]_{\mathcal{B}} = g^*(f) = \frac{\int_\Omega f\bar{g}|g|^{p-2}d\mu}{\|g\|^{p-2}_{L^p(\Omega,\mu)}}.$$

With the above preparation, we shall study a special kind of RKBS which have desired properties.

4.2 S.i.p. RKBS

Let X be a prescribed input space. We call a uniformly convex and uniformly Fréchet differentiable RKBS on X an **s.i.p. reproducing kernel Banach space** (**s.i.p. RKBS**). Again, we see immediately that an RKHS is an s.i.p. RKBS. Also, the dual of an s.i.p. RKBS remains an s.i.p. RKBS. An s.i.p. RKBS $\mathcal{B}$ is by definition uniformly Fréchet differentiable. Therefore, it has a unique semi-inner-product, which by Lemma 8 represents all the interaction between $\mathcal{B}$ and $\mathcal{B}^*$. This leads to a more specific representation of the reproducing kernel. Precisely, we have the following consequences.

Theorem 9 *Let $\mathcal{B}$ be an s.i.p. RKBS on X and K its reproducing kernel. Then there exists a unique function $G : X \times X \to \mathbb{C}$ such that $\{G(x,\cdot) : x \in X\} \subseteq \mathcal{B}$ and*

$$f(x) = [f, G(x,\cdot)]_{\mathcal{B}}, \quad \text{for all } f \in \mathcal{B},\ x \in X. \tag{30}$$

Moreover, there holds the relationship

$$K(\cdot,x) = (G(x,\cdot))^*, \ \ x \in X \tag{31}$$

and

$$f^*(x) = [K(x,\cdot), f]_{\mathcal{B}}, \quad \text{for all } f \in \mathcal{B},\ x \in X. \tag{32}$$

Proof By Lemma 8, for each $x \in X$ there exists a function $G_x \in \mathcal{B}$ such that $f(x) = [f, G_x]_{\mathcal{B}}$ for all $f \in \mathcal{B}$. We define $G : X \times X \to \mathbb{C}$ by $G(x,y) := G_x(y)$, $x, y \in X$. We see that $G(x,\cdot) = G_x \in \mathcal{B}$, $x \in X$, and there holds (30). By the uniqueness in the Riesz representation theorem, such a function G is unique. To prove the remaining claims, we recall from Theorem 2 that the reproducing kernel K satisfies for each $f \in \mathcal{B}$ that

$$f(x) = (f, K(\cdot,x))_{\mathcal{B}}, \ \ x \in X. \tag{33}$$

and

$$f^*(x) = (K(x,\cdot), f^*)_{\mathcal{B}}, \ \ x \in X. \tag{34}$$

By (25), (30) and (33), we have for each $x \in X$ that

$$(f, (G(x,\cdot))^*)_{\mathcal{B}} = [f, G(x,\cdot)]_{\mathcal{B}} = f(x) = (f, K(\cdot,x))_{\mathcal{B}}, \ \ f \in \mathcal{B}.$$

The above equation implies (31). Equation (25) also implies that

$$(K(x,\cdot), f^*)_{\mathcal{B}} = [K(x,\cdot), f]_{\mathcal{B}}.$$

This together with equation (34) proves (32) and completes the proof. ■

We call the unique function G in Theorem 9 the **s.i.p. kernel** of the s.i.p. RKBS $\mathcal{B}$. It coincides with the reproducing kernel K when $\mathcal{B}$ is an RKHS. In general, when $G = K$ in Theorem 9, we call G an **s.i.p. reproducing kernel**. By (30), an s.i.p. reproducing kernel G satisfies the following generalization of (3)

$$G(x,y) = [G(x,\cdot), G(y,\cdot)]_{\mathcal{B}}, \ \ x, y \in X. \tag{35}$$

We shall give a characterization of an s.i.p. reproducing kernel in terms of its corresponding feature map. To this end, for a mapping Φ from X to a uniformly convex and uniformly Fréchet differentiable Banach space $\mathcal{W}$, we denote by Φ^* the mapping from X to $\mathcal{W}^*$ defined as

$$\Phi^*(x) := (\Phi(x))^*, \ \ x \in X.$$

Theorem 10 *Let $\mathcal{W}$ be a uniformly convex and uniformly Fréchet differentiable Banach space and Φ a mapping from X to $\mathcal{W}$ such that*

$$\overline{\text{span}}\,\Phi(X) = \mathcal{W}, \ \ \overline{\text{span}}\,\Phi^*(X) = \mathcal{W}^*. \tag{36}$$

Then $\mathcal{B} := \{[u, \Phi(\cdot)]_{\mathcal{W}} : u \in \mathcal{W}\}$ equipped with

$$\Big[[u, \Phi(\cdot)]_{\mathcal{W}}, [v, \Phi(\cdot)]_{\mathcal{W}}\Big]_{\mathcal{B}} := [u, v]_{\mathcal{W}} \tag{37}$$

and $\mathcal{B}^* := \{[\Phi(\cdot), u]_{\mathcal{W}} : u \in \mathcal{W}\}$ with

$$\Big[[\Phi(\cdot), u]_{\mathcal{W}}, [\Phi(\cdot), v]_{\mathcal{W}}\Big]_{\mathcal{B}^*} := [v, u]_{\mathcal{W}}$$

are uniformly convex and uniformly Fréchet differentiable Banach spaces. And $\mathcal{B}^*$ is the dual of $\mathcal{B}$ with the bilinear form

$$\Big([u, \Phi(\cdot)]_{\mathcal{W}}, [\Phi(\cdot), v]_{\mathcal{W}}\Big)_{\mathcal{B}} := [u, v]_{\mathcal{W}}, \ \ u, v \in \mathcal{W}. \tag{38}$$

Moreover, the s.i.p. kernel G of $\mathcal{B}$ is given by

$$G(x, y) = [\Phi(x), \Phi(y)]_{\mathcal{W}}, \ \ x, y \in X, \tag{39}$$

which coincides with its reproducing kernel K.

Proof We shall show (39) only. The other results can be proved using arguments similar to those in Theorem 3 and those in the proof of Theorem 7 in Giles (1967). Let $f \in \mathcal{B}$. Then there exists a unique $u \in \mathcal{W}$ such that $f = [u, \Phi(\cdot)]_{\mathcal{W}}$. By (38), for $y \in X$,

$$f(y) = [u, \Phi(y)]_{\mathcal{W}} = ([u, \Phi(\cdot)]_{\mathcal{W}}, [\Phi(\cdot), \Phi(y)]_{\mathcal{W}})_{\mathcal{B}} = (f, [\Phi(\cdot), \Phi(y)]_{\mathcal{W}})_{\mathcal{B}}.$$

Comparing the above equation with (33), we obtain that

$$K(\cdot, y) = [\Phi(\cdot), \Phi(y)]_{\mathcal{W}}. \tag{40}$$

On the other hand, by (37), for $x \in X$

$$f(x) = [u, \Phi(x)]_{\mathcal{W}} = [[u, \Phi(\cdot)]_{\mathcal{W}}, [\Phi(x), \Phi(\cdot)]_{\mathcal{W}}]_{\mathcal{B}},$$

which implies that the s.i.p. kernel of $\mathcal{B}$ is

$$G(x, \cdot) = [\Phi(x), \Phi(\cdot)]_{\mathcal{W}}. \tag{41}$$

By (40) and (41),

$$K(x, y) = G(x, y) = [\Phi(x), \Phi(y)]_{\mathcal{W}},$$

which completes the proof. ■

As a direct consequence of the above theorem, we have the following characterization of s.i.p. reproducing kernels.

Theorem 11 *A function G on $X \times X$ is an s.i.p. reproducing kernel if and only if it is of the form (39), where Φ is a mapping from X to a uniformly convex and uniformly Fréchet differentiable Banach space $\mathcal{W}$ satisfying (36).*

Proof The sufficiency is implied by Theorem 10. For the necessity, suppose that G is an s.i.p. reproducing kernel for some s.i.p. RKBS $\mathcal{B}$ on X. We choose $\mathcal{W} = \mathcal{B}$ and $\Phi(x) := G(x, \cdot)$. Then G has the form (39) by equation (35). Moreover, by (8), $\operatorname{span}\Phi(X)$ is dense in $\mathcal{W}$. Assume that $\operatorname{span}\Phi^*(X)$ is not dense in $\mathcal{W}^*$. Then by the Hahn-Banach theorem and Lemma 8, there exists a nontrivial $f \in \mathcal{B}$ such that $[\Phi^*(x), f^*]_{\mathcal{B}^*} = 0, x \in X$. Thus, by (28) we get that

$$f(x) = [f, G(x, \cdot)]_{\mathcal{B}} = [f, \Phi(x)]_{\mathcal{W}} = [\Phi^*(x), f^*]_{\mathcal{B}^*} = 0, \ \ x \in X.$$

We end up with a zero function f, a contradiction. The proof is complete. ■

The mapping Φ and space $\mathcal{W}$ in the above theorem will be called a **feature map** and **feature space** of the s.i.p. reproducing kernel G, respectively.

By the duality relation (31) and the denseness condition (9), the s.i.p kernel G of an s.i.p. RKBS $\mathcal{B}$ on X satisfies

$$\overline{\operatorname{span}}\{(G(x, \cdot))^* : x \in X\} = \mathcal{B}^*. \tag{42}$$

It is also of the form (35). By Theorem 11, G is identical with the reproducing kernel K for $\mathcal{B}$ if and only if

$$\overline{\operatorname{span}}\{G(x, \cdot) : x \in X\} = \mathcal{B}. \tag{43}$$

If $\mathcal{B}$ is not a Hilbert space then the duality mapping from $\mathcal{B}$ to $\mathcal{B}^*$ is nonlinear. Thus, it may not preserve the denseness of a linear span. As a result, (43) would not follow automatically from (42). Here we remark that for most finite dimensional s.i.p. RKBS, (42) implies (43). This is due to the well-known fact that for all $n \in \mathbb{N}$, the set of $n \times n$ singular matrices has Lebesgue measure zero in $\mathbb{C}^{n \times n}$. Therefore, the s.i.p. kernel for most finite dimensional s.i.p. RKBS is the same as the reproducing kernel. Nevertheless, we shall give an explicit example to illustrate that the two kernels might be different.

For each $p \in (1, +\infty)$ and $n \in \mathbb{N}$, we denote by $\ell^p(\mathbb{N}_n)$ the Banach space of vectors in $\mathbb{C}^n$ with norm

$$\|a\|_{\ell^p(\mathbb{N}_n)} := \left(\sum_{j \in \mathbb{N}_n} |a_j|^p \right)^{1/p}, \ \ a = (a_j : j \in \mathbb{N}_n) \in \mathbb{C}^n.$$

As pointed out at the end of Section 4.1, $\ell^p(\mathbb{N}_n)$ is uniformly convex and uniformly Fréchet differentiable. Its dual space is $\ell^q(\mathbb{N}_n)$. To construct the example, we introduce three vectors in $\ell^3(\mathbb{N}_3)$ by setting

$$e_1 := (2, 9, 1), \ e_2 := (1, 8, 0), \ e_3 := (5, 5, 3).$$

By (29), their dual elements in $\ell^{\frac{3}{2}}(\mathbb{N}_3)$ are

$$e_1^* = \frac{1}{(738)^{1/3}}(4, 81, 1), \ e_2^* = \frac{1}{(513)^{1/3}}(1, 64, 0), \ e_3^* = \frac{1}{(277)^{1/3}}(25, 25, 9).$$

It can be directly verified that $\{e_1, e_2, e_3\}$ is linearly independent but $\{e_1^*, e_2^*, e_3^*\}$ is not. Therefore,

$$\operatorname{span}\{e_1, e_2, e_3\} = \ell^3(\mathbb{N}_3) \tag{44}$$

while

$$\operatorname{span}\{e_1^*, e_2^*, e_3^*\} \subsetneqq \ell^{\frac{3}{2}}(\mathbb{N}_3). \tag{45}$$

With the above preparations, we let $\mathbb{N}_3$ be the input space, Φ the function from $\mathbb{N}_3$ to $\ell^3(\mathbb{N}_3)$ defined by $\Phi(j) = e_j$, $j \in \mathbb{N}_3$, and $\mathcal{B}$ the space of all the functions $\Phi_u := [\Phi(\cdot), u]_{\ell^3(\mathbb{N}_3)}$, $u \in \ell^3(\mathbb{N}_3)$, on $\mathbb{N}_3$. By equation (44),

$$\|\Phi_u\| := \|u\|_{\ell^3(\mathbb{N}_3)}, \ \ u \in \ell^3(\mathbb{N}_3)$$

defines a norm on $\mathcal{B}$. It is clear that point evaluations are continuous on $\mathcal{B}$ under this norm. Furthermore, since the linear mapping $\Phi_u \to u^*$ is isometrically isomorphic from $\mathcal{B}$ to $\ell^{\frac{3}{2}}(\mathbb{N}_3)$, $\mathcal{B}$ is a uniformly convex and uniformly Fréchet differentiable Banach space. By this fact, we obtain that $\mathcal{B}$ is an s.i.p. RKBS with semi-inner-product

$$[\Phi_u, \Phi_v]_{\mathcal{B}} = [v, u]_{\ell^3(\mathbb{N}_3)}, \ \ u, v \in \ell^3(\mathbb{N}_3). \tag{46}$$

The above equation implies that the s.i.p. kernel G for $\mathcal{B}$ is

$$G(j,k) = [e_k, e_j]_{\ell^3(\mathbb{N}_3)}, \ \ j,k \in \mathbb{N}_3. \tag{47}$$

Recall that the reproducing kernel K for $\mathcal{B}$ satisfies the denseness condition (8). Consequently, to show that $G \neq K$, it suffices to show that

$$\operatorname{span}\{G(j,\cdot) : j \in \mathbb{N}_3\} \subsetneqq \mathcal{B}. \tag{48}$$

To this end, we notice by (45) that there exists a nonzero element $v \in \ell^3(\mathbb{N}_3)$ such that

$$[v, e_j]_{\ell^3(\mathbb{N}_3)} = (v, e_j^*)_{\ell^3(\mathbb{N}_3)} = 0, \ \ j \in \mathbb{N}_3.$$

As a result, the nonzero function Φ_v satisfies by (46) and (47) that

$$[G(j,\cdot), \Phi_v]_{\mathcal{B}} = [\Phi_{e_j}, \Phi_v]_{\mathcal{B}} = [v, e_j]_{\ell^3(\mathbb{N}_3)} = 0, \ \ j \in \mathbb{N}_3,$$

which proves (48), and implies that the s.i.p. kernel and reproducing kernel for $\mathcal{B}$ are different. By (45), this is essentially due to the reason that the second condition of (36) is not satisfied.

4.3 Properties of S.i.p. Reproducing Kernels

The existence of a semi-inner-product makes it possible to study properties of RKBS and their reproducing kernels. For illustration, we present below three of these properties.

4.3.1 NON-POSITIVE DEFINITENESS

An $n \times n$ matrix M over a number field $\mathbb{F}$ ($\mathbb{C}$ or $\mathbb{R}$) is said to be positive semi-definite if for all $(c_j : j \in \mathbb{N}_n) \in \mathbb{F}^n$

$$\sum_{j \in \mathbb{N}_n} \sum_{k \in \mathbb{N}_n} c_j \overline{c_k} M_{jk} \geq 0.$$

We shall consider positive semi-definiteness of matrices $G[\mathbf{x}]$ as defined in (1) for an s.i.p. reproducing kernel G on X.

Let $\Phi : X \to \mathcal{W}$ be a feature map for G, that is, (39) and (36) hold. By properties 3 and 4 in the definition of a semi-inner-product, we have that

$$G(x,x) \geq 0, \ \ x \in X \tag{49}$$

and

$$|G(x,y)|^2 \leq G(x,x)G(y,y), \ \ x,y \in X. \tag{50}$$

Notice that if a complex matrix is positive semi-definite then it must be hermitian. Since a semi-inner-product is in general not an inner product, we can not expect a complex s.i.p. kernel to be positive definite. In the real case, inequalities (49) and (50) imply that $G[\mathbf{x}]$ is positive semi-definite for all $\mathbf{x} \subseteq X$ with cardinality less than or equal to two. However, $G[\mathbf{x}]$ might stop being positive semi-definite if $\mathbf{x}$ contains more than two points. We shall give an explicit example to explain this phenomenon.

Set $p \in (1,+\infty)$ and $\mathcal{W} := \ell^p(\mathbb{N}_2)$. We let $X := \mathbb{R}_+ := [0,+\infty)$ and $\Phi(x) = (1,x), x \in X$. Thus,

$$\Phi^*(x) = \frac{(1,x^{p-1})}{(1+x^p)^{\frac{p-2}{p}}}, \ \ x \in X.$$

Clearly, Φ satisfies the denseness condition (36). The corresponding s.i.p. reproducing kernel G is constructed as

$$G(x,y) = [\Phi(x),\Phi(y)]_{\mathcal{W}} = \frac{1+xy^{p-1}}{(1+y^p)^{\frac{p-2}{p}}}, \ \ x,y \in X. \tag{51}$$

Proposition 12 *For the s.i.p. reproducing kernel G defined by (51), matrix $G[\mathbf{x}]$ is positive semi-definite for all $\mathbf{x} = \{x,y,z\} \subseteq X$ if and only if $p = 2$.*

Proof If $p = 2$ then $\mathcal{W}$ is a Hilbert space. As a result, G is a positive definite kernel. Hence, for all finite subsets $\mathbf{x} \subseteq X$, $G[\mathbf{x}]$ is positive semi-definite.

Assume that $G[\mathbf{x}]$ is positive semi-definite for all $\mathbf{x} = \{x,y,z\} \subseteq X$. Choose $\mathbf{x} := \{0,1,t\}$ where t is a positive number to be specified later. Then we have by (51) that

$$G[\mathbf{x}] = \begin{bmatrix} 1 & 2^{2/p-1} & \dfrac{1}{(1+t^p)^{1-2/p}} \\ 1 & 2^{2/p} & \dfrac{1+t^{p-1}}{(1+t^p)^{1-2/p}} \\ 1 & \dfrac{1+t}{2^{1-2/p}} & \dfrac{1+t^p}{(1+t^p)^{1-2/p}} \end{bmatrix}.$$

Let M be the symmetrization of $G[\mathbf{x}]$ given as

$$M = \begin{bmatrix} 1 & \dfrac{1}{2}+2^{2/p-2} & \dfrac{1}{2}+\dfrac{1}{2(1+t^p)^{1-2/p}} \\ \dfrac{1}{2}+2^{2/p-2} & 2^{2/p} & \dfrac{1+t}{2^{2-2/p}}+\dfrac{1+t^{p-1}}{2(1+t^p)^{1-2/p}} \\ \dfrac{1}{2}+\dfrac{1}{2(1+t^p)^{1-2/p}} & \dfrac{1+t}{2^{2-2/p}}+\dfrac{1+t^{p-1}}{2(1+t^p)^{1-2/p}} & \dfrac{1+t^p}{(1+t^p)^{1-2/p}} \end{bmatrix}.$$

Matrix M preserves the positive semi-definiteness of $G[\mathbf{x}]$. Therefore, its determinant $|M|$ must be nonnegative. Through an analysis of the asymptotic behavior of the component of M as t goes to infinity, we obtain that

$$|M| = -\frac{t^2}{8}\left(2^{\frac{2}{p}} - 2\right)^2 + \varphi(t), \;\; t > 0,$$

where φ is a function satisfying that

$$\lim_{t\to\infty} \frac{\varphi(t)}{t^2} = 0.$$

Therefore, $|M|$ being always nonnegative forces $2^{\frac{2}{p}} - 2 = 0$, which occurs only if $p = 2$. ■

By Proposition 12, non-positive semi-definiteness is a characteristic of s.i.p. reproducing kernels for RKBS that distincts them from reproducing kernels for RKHS.

4.3.2 POINTWISE CONVERGENCE

If f_n converges to f in an s.i.p. RKBS with its s.i.p. kernel G then $f_n(x)$ converges to $f(x)$ for any $x \in X$ and the limit is uniform on the set where $G(x,x)$ is bounded. This follows from (30) and the Cauchy-Schwartz inequality by

$$|f_n(x) - f(x)| = |[f_n - f, G(x,\cdot)]_{\mathcal{B}}| \leq \|f_n - f\|_{\mathcal{B}} \sqrt{[G(x,\cdot), G(x,\cdot)]_{\mathcal{B}}} = \sqrt{G(x,x)}\|f_n - f\|_{\mathcal{B}}.$$

4.3.3 WEAK UNIVERSALITY

Suppose that X is metric space and G is an s.i.p. reproducing kernel on X. We say that G is **universal** if G is continuous on $X \times X$ and for all compact subsets $\mathcal{Z} \subseteq X$, $\operatorname{span}\{G(x,\cdot) : x \in \mathcal{Z}\}$ is dense in $C(\mathcal{Z})$ (Micchelli et al., 2006; Steinwart, 2001). Universality of a kernel ensures that it can approximate any continuous target function uniformly on compact subsets of the input space. This is crucial for the consistence of the learning algorithms with the kernel. We shall discuss the case when X is itself a compact metric space. Here we are concerned with the ability of G to approximate any continuous target function on X uniformly. For this purpose, we call a continuous kernel G on a compact metric space X **weakly universal** if $\operatorname{span}\{G(x,\cdot) : x \in X\}$ is dense in $C(X)$. We shall present a characterization of weak universality. The results in the cases of positive definite kernels and vector-valued positive definite kernels have been established respectively in Micchelli et al. (2006) and Caponnetto et al. (2008).

Proposition 13 *Let Φ be a feature map from a compact metric space X to $\mathcal{W}$ such that both $\Phi : X \to \mathcal{W}$ and $\Phi^* : X \to \mathcal{W}^*$ are continuous. Then the s.i.p. reproducing kernel G defined by (39) is continuous on $X \times X$, and there holds in $C(X)$ the equality of subspaces*

$$\overline{\operatorname{span}}\{G(x,\cdot) : x \in X\} = \overline{\operatorname{span}}\{[u, \Phi(\cdot)]_{\mathcal{W}} : u \in \mathcal{W}\}.$$

Consequently, G is weakly universal if and only if

$$\overline{\operatorname{span}}\{[u, \Phi(\cdot)]_{\mathcal{W}} : u \in \mathcal{W}\} = C(X).$$

Proof First, we notice by

$$G(x,y) = [\Phi(x), \Phi(y)]_{\mathcal{W}} = (\Phi(x), \Phi^*(y))_{\mathcal{W}}, \;\; x,y \in X$$

that G is continuous on $X \times X$. Similarly, for each $u \in \mathcal{W}$, $[u, \Phi(\cdot)]_{\mathcal{W}} = (u, \Phi^*(\cdot))_{\mathcal{W}} \in C(X)$. Now since

$$G(x,\cdot) = [\Phi(x), \Phi(\cdot)]_{\mathcal{W}} \in \{[u, \Phi(\cdot)]_{\mathcal{W}} : u \in \mathcal{W}\},$$

we have the inclusion

$$\overline{\text{span}}\{G(x,\cdot) : x \in X\} \subseteq \overline{\text{span}}\{[u, \Phi(\cdot)]_{\mathcal{W}} : u \in \mathcal{W}\}.$$

On the other hand, let $u \in \mathcal{W}$. By denseness condition (36), there exists a sequence $v_n \in \text{span}\{\Phi(x) : x \in X\}$ that converges to u. Since G is continuous on the compact space $X \times X$, it is bounded. Thus, by the property of pointwise convergence discussed before, $[v_n, \Phi(\cdot)]_{\mathcal{W}}$ converges in $C(X)$ to $[u, \Phi(\cdot)]_{\mathcal{W}}$. Noting that

$$[v_n, \Phi(\cdot)]_{\mathcal{W}} \in \text{span}\{G(x,\cdot) : x \in X\}, \;\; n \in \mathbb{N},$$

we have the reverse inclusion

$$\overline{\text{span}}\{[u, \Phi(\cdot)]_{\mathcal{W}} : u \in \mathcal{W}\} \subseteq \overline{\text{span}}\{G(x,\cdot) : x \in X\},$$

which proves the result. ■

We remark that in the case that $\mathcal{W}$ is a Hilbert space, the idea in the above proof can be applied to show with less effort the main result in Caponnetto et al. (2008) and Micchelli et al. (2006), that is, for each compact subset $\mathcal{Z} \subseteq X$

$$\overline{\text{span}}\{G(x,\cdot) : x \in \mathcal{Z}\} = \overline{\text{span}}\{[u, \Phi(\cdot)]_{\mathcal{W}} : u \in \mathcal{W}\},$$

where the two closures are taken in $C(\mathcal{Z})$. A key element missing in the Banach space is the orthogonal decomposition in a Hilbert space $\mathcal{W}$:

$$\mathcal{W} = (\overline{\text{span}}\Phi(\mathcal{Z})) \oplus \Phi(\mathcal{Z})^{\perp}, \;\; \mathcal{Z} \subseteq X.$$

For a normed vector space V, we denote for each $A \subseteq V$ by $A^{\perp} := \{v^* \in V^* : (a, v^*)_V = 0, \; a \in A\}$, and for each $B \subseteq V^*$, ${}^{\perp}B := \{a \in V : (a, v^*)_V = 0, \; v^* \in B\}$. A Banach space $\mathcal{W}$ is in general not the direct sum of $\overline{\text{span}}\Phi(\mathcal{Z})$ and ${}^{\perp}\Phi^*(\mathcal{Z})$. In fact, closed subspaces in $\mathcal{W}$ may not always have an algebraic complement unless $\mathcal{W}$ is isomorphic to a Hilbert space (see, Conway, 1990, page 94).

Universality and other properties of s.i.p. reproducing kernels will be treated specially in future work. One of the main purposes of this study is to apply the tool of s.i.p. reproducing kernels to learning in Banach spaces. To be precise, we shall develop in the framework of s.i.p. RKBS several standard learning schemes.

5. Representer Theorems for Standard Learning Schemes

In this section, we assume that $\mathcal{B}$ is an s.i.p. RKBS on X with the s.i.p. reproducing kernel G defined by a feature map $\Phi : X \to \mathcal{W}$ as in (39). We shall develop in this framework several standard learning schemes including minimal norm interpolation, regularization network, support vector machines, and kernel principal component analysis. For introduction and discussions of these widely used algorithms in RKHS, see, for example, Cucker and Smale (2002), Evgeniou et al. (2000), Micchelli and Pontil (2005), Schölkopf and Smola (2002), Shawe-Taylor and Cristianini (2004) and Vapnik (1998).

5.1 Minimal Norm Interpolation

The minimal norm interpolation is to find, among all functions in $\mathcal{B}$ that interpolate a prescribed set of points, a function with the minimal norm. Let $\mathbf{x} := \{x_j : j \in \mathbb{N}_n\}$ be a fixed finite set of distinct points in X and set for each $\mathbf{y} := (y_j : j \in \mathbb{N}_n) \in \mathbb{C}^n$

$$I_{\mathbf{y}} := \{f \in \mathcal{B} : f(x_j) = y_j,\ j \in \mathbb{N}_n\}.$$

Our purpose is to find $f_0 \in I_{\mathbf{y}}$ such that

$$\|f_0\|_{\mathcal{B}} = \inf\{\|f\|_{\mathcal{B}} : f \in I_{\mathbf{y}}\} \tag{52}$$

provided that I_y is nonempty. Our first concern is of course the condition ensuring that $I_{\mathbf{y}}$ is nonempty. To address this issue, let us recall the useful property of the s.i.p. reproducing kernel G:

$$f(x) = [f, G(x,\cdot)]_{\mathcal{B}} = [G(\cdot,x), f^*]_{\mathcal{B}^*},\ \ x \in X,\ f \in \mathcal{B}. \tag{53}$$

Lemma 14 *The set $I_{\mathbf{y}}$ is nonempty for any $\mathbf{y} \in \mathbb{C}^n$ if and only if $G_{\mathbf{x}} := \{G(\cdot,x_j) : j \in \mathbb{N}_n\}$ is linearly independent in $\mathcal{B}^*$.*

Proof Observe that $I_{\mathbf{y}}$ is nonempty for any $\mathbf{y} \in \mathbb{C}^n$ if and only if $\operatorname{span}\{(f(x_j) : j \in \mathbb{N}_n) : f \in \mathcal{B}\}$ is dense in $\mathbb{C}^n$. Using the reproducing property (53), we have for each $(c_j : j \in \mathbb{N}_n) \in \mathbb{C}^n$ that

$$\sum_{j\in\mathbb{N}_n} c_j f(x_j) = \sum_{j\in\mathbb{N}_n} c_j [f, G(x_j,\cdot)]_{\mathcal{B}} = \left[\sum_{j\in\mathbb{N}_n} c_j G(\cdot,x_j), f^*\right]_{\mathcal{B}^*}.$$

Thus,

$$\sum_{j\in\mathbb{N}_n} c_j f(x_j) = 0, \quad \text{for all } f \in \mathcal{B}$$

if and only if

$$\sum_{j\in\mathbb{N}_n} c_j G(\cdot,x_j) = 0.$$

This implies that $G_{\mathbf{x}}$ is linearly independent in $\mathcal{B}^*$ if and only if $\operatorname{span}\{(f(x_j) : j \in \mathbb{N}_n) : f \in \mathcal{B}\}$ is dense in $\mathbb{C}^n$. Therefore, $I_{\mathbf{y}}$ is nonempty for any $\mathbf{y} \in \mathbb{C}^n$ if and only if $G_{\mathbf{x}}$ is linearly independent. ■

We next show that the minimal norm interpolation problem in $\mathcal{B}$ always has a unique solution under the hypothesis that $G_{\mathbf{x}}$ is linearly independent. The following useful property of a uniformly convex Banach space is crucial (see, for example, Istrăţescu, 1984, page 53).

Lemma 15 *If V is a uniformly convex Banach space, then for any nonempty closed convex subset $A \subseteq V$ and any $x \in V$ there exists a unique $x_0 \in A$ such that*

$$\|x - x_0\|_V = \inf\{\|x - a\|_V : a \in A\}.$$

Proposition 16 *If $G_{\mathbf{x}}$ is linearly independent in $\mathcal{B}^*$, then for any $\mathbf{y} \in \mathbb{C}^n$ there exists a unique $f_0 \in I_{\mathbf{y}}$ satisfying (52).*

Proof By Lemma 14, $I_{\mathbf{y}}$ is nonempty. Note also that it is closed and convex. Since $\mathcal{B}$ is uniformly convex, by Lemma 15, there exists a unique $f_0 \in I_{\mathbf{y}}$ such that

$$\|f_0\|_{\mathcal{B}} = \|0 - f_0\|_{\mathcal{B}} = \inf\{\|0 - f\|_{\mathcal{B}} = \|f\|_{\mathcal{B}} : f \in I_{\mathbf{y}}\}.$$

The above equation proves the result. ■

We shall establish a representation of the minimal norm interpolator f_0. For this purpose, a simple observation is made based on the following fact connecting orthogonality with best approximation in s.i.p. spaces (Giles, 1967).

Lemma 17 *If V is a uniformly Fréchet differentiable normed vector space, then $\|x + \lambda y\|_V \geq \|x\|_V$ for all $\lambda \in \mathbb{C}$ if and only if $[y, x]_V = 0$.*

Lemma 18 *If $I_{\mathbf{y}}$ is nonempty then $f_0 \in I_{\mathbf{y}}$ is the minimizer of (52) if and only if*

$$[g, f_0]_{\mathcal{B}} = 0, \quad \text{for all } g \in I_0. \tag{54}$$

Proof Let $f_0 \in I_{\mathbf{y}}$. It is obvious that f_0 is the minimizer of (52) if and only if

$$\|f_0 + g\|_{\mathcal{B}} \geq \|f_0\|_{\mathcal{B}}, \quad g \in I_0.$$

Since I_0 is a linear subspace of $\mathcal{B}$, the result follows immediately from Lemma 17. ■

The following result is of the representer theorem type. For the representer theorem in learning with positive definite kernels, see, for example, Argyriou et al. (2008), Kimeldorf and Wahba (1971) and Schölkopf et al. (2001).

Theorem 19 *(Representer Theorem) Suppose that $G_{\mathbf{x}}$ is linearly independent in $\mathcal{B}^*$ and f_0 is the solution of the minimal norm interpolation (52). Then there exists $\mathbf{c} = (c_j : j \in \mathbb{N}_n) \in \mathbb{C}^n$ such that*

$$f_0^* = \sum_{j \in \mathbb{N}_n} c_j G(\cdot, x_j). \tag{55}$$

Moreover, a function of the form in the right hand side above is the solution if and only if $\mathbf{c}$ satisfies

$$\left[G(\cdot, x_k), \sum_{j \in \mathbb{N}_n} c_j G(\cdot, x_j) \right]_{\mathcal{B}^*} = y_k, \quad k \in \mathbb{N}_n. \tag{56}$$

Proof Note that (54) is equivalent to $f_0^* \in I_0{}^\perp$ and that $I_0 = {}^\perp G_{\mathbf{x}}$. Therefore, f_0 satisfies (54) if and only if

$$f_0^* \in ({}^\perp G_{\mathbf{x}})^\perp.$$

Recall a consequence of the Hahn-Banach theorem that in a reflexive Banach space $\mathcal{B}$, for each $B \subseteq \mathcal{B}^*$,

$$({}^\perp B)^\perp = \overline{\operatorname{span}} B. \tag{57}$$

By this fact, (55) holds true for some $\mathbf{c} \in \mathbb{C}^n$.

Suppose that $f \in \mathcal{B}$ is of the form $f^* = \sum_{j\in\mathbb{N}_n} c_j G(\cdot, x_j)$ where $\mathbf{c}$ satisfies (56). Then $f^* \in \operatorname{span} G_{\mathbf{x}}$. By (57), f^* satisfies (54). Furthermore, (56) implies that

$$f(x_k) = [f, G(x_k,\cdot)]_{\mathcal{B}} = [G(\cdot,x_k), f^*]_{\mathcal{B}^*} = \left[G(\cdot,x_k), \sum_{j\in\mathbb{N}_n} c_j G(\cdot,x_j)\right]_{\mathcal{B}^*} = y_k, \ \ k \in \mathbb{N}_n. \tag{58}$$

That is, $f \in I_{\mathbf{y}}$. By Lemma 18, $f = f_0$. On the other hand, f_0 has the form (55). As shown in (58), (56) is simply the interpolation condition that $f_0(x_k) = y_k, k \in \mathbb{N}_n$. Thus, it must be true. The proof is complete. ■

Applying the inverse of the duality mapping to both sides of (55) yields a representation of f_0 in the space $\mathcal{B}$. However, since the duality mapping is nonadditive unless $\mathcal{B}$ is an RKHS, this procedure in general does not result in a linear representation.

We conclude that under the condition that $G_{\mathbf{x}}$ is linearly independent, the minimal norm interpolation problem (52) has a unique solution, and finding the solution reduces to solving the system (56) of equations about $\mathbf{c} \in \mathbb{C}^n$. The solution $\mathbf{c}$ of (56) is unique by Theorem 19. Again, the difference from the result for RKHS is that (56) is often nonlinear about $\mathbf{c}$ since by Proposition 6 a semi-inner-product is generally nonadditive about the second variable.

To see an explicit form of (56), we shall reformulate it in terms of the feature map Φ from X to $\mathcal{W}$. Let $\mathcal{B}$ and $\mathcal{B}^*$ be identified as in Theorem 10. Then (56) has the equivalent form

$$\left[\Phi^*(x_k), \sum_{j\in\mathbb{N}_n} c_j \Phi^*(x_j)\right]_{\mathcal{B}^*} = y_k, \ \ k \in \mathbb{N}_n.$$

In the particular case that $\mathcal{W} = L^p(\Omega,\mu)$, $p \in (1,+\infty)$ on some measure space $(\Omega, \mathcal{F}, \mu)$, and $\mathcal{W}^* = L^q(\Omega,\mu)$, the above equation is rewritten as

$$\int_\Omega \Phi^*(x_k) \overline{\sum_{j\in\mathbb{N}_n} c_j \Phi^*(x_j)} \left|\sum_{j\in\mathbb{N}_n} c_j \Phi^*(x_j)\right|^{q-2} d\mu = y_k \left\|\sum_{j\in\mathbb{N}_n} c_j \Phi^*(x_j)\right\|_{L^q(\Omega,\mu)}^{q-2}, \ \ k \in \mathbb{N}_n.$$

5.2 Regularization Network

We consider learning a predictor function from a finite sample data $\mathbf{z} := \{(x_j, y_j) : j \in \mathbb{N}_n\} \subseteq X \times \mathbb{C}$ in this subsection. The predictor function will yield from a regularized learning algorithm. Let $\mathcal{L} : \mathbb{C} \times \mathbb{C} \to \mathbb{R}_+$ be a *loss function* that is continuous and convex with respect to its first variable. For each $f \in \mathcal{B}$, we set

$$\mathcal{E}_{\mathbf{z}}(f) := \sum_{j\in\mathbb{N}_n} \mathcal{L}(f(x_j), y_j) \text{ and } \mathcal{E}_{\mathbf{z},\mu}(f) := \mathcal{E}_{\mathbf{z}}(f) + \mu \|f\|_{\mathcal{B}}^2,$$

where μ is a positive regularization parameter. The predictor function that we learn from the sample data $\mathbf{z}$ will be the function f_0 satisfying

$$\mathcal{E}_{\mathbf{z},\mu}(f_0) = \inf\{\mathcal{E}_{\mathbf{z},\mu}(f) : f \in \mathcal{B}\}. \tag{59}$$

One can always make this choice as we shall prove that the minimizer of (59) exists and is unique.

Theorem 20 *There exists a unique $f_0 \in \mathcal{B}$ satisfying (59).*

Proof We first show the existence. If $f \in \mathcal{B}$ satisfies $\|f\|_{\mathcal{B}}^2 > \frac{1}{\mu}\mathcal{E}_{\mathbf{z},\mu}(0)$ then

$$\mathcal{E}_{\mathbf{z},\mu}(f) \geq \mu\|f\|_{\mathcal{B}}^2 > \mathcal{E}_{\mathbf{z},\mu}(0).$$

Thus

$$\inf\{\mathcal{E}_{\mathbf{z},\mu}(f) : f \in \mathcal{B}\} = \inf\left\{\mathcal{E}_{\mathbf{z},\mu}(f) : f \in \mathcal{B},\ \|f\|_{\mathcal{B}}^2 \leq \frac{1}{\mu}\mathcal{E}_{\mathbf{z},\mu}(0)\right\}.$$

Let

$$e := \inf\left\{\mathcal{E}_{\mathbf{z},\mu}(f) : f \in \mathcal{B},\ \|f\|_{\mathcal{B}}^2 \leq \frac{1}{\mu}\mathcal{E}_{\mathbf{z},\mu}(0)\right\}$$

and

$$A := \left\{f \in \mathcal{B} : \|f\|_{\mathcal{B}}^2 \leq \frac{1}{\mu}\mathcal{E}_{\mathbf{z},\mu}(0)\right\}.$$

Then, there exists a sequence $f_k \in A$, $k \in \mathbb{N}$, such that

$$e \leq \mathcal{E}_{\mathbf{z},\mu}(f_k) \leq e + \frac{1}{k}. \tag{60}$$

Since $\mathcal{B}$ is reflexive, A is weakly compact, that is, we may assume that there exists $f_0 \in A$ such that for all $g \in \mathcal{B}$

$$\lim_{k\to\infty}[f_k, g]_{\mathcal{B}} = [f_0, g]_{\mathcal{B}}. \tag{61}$$

In particular, the choice $g := G(x_j, \cdot)$, $j \in \mathbb{N}_n$ yields that $f_k(x_j)$ converges to $f_0(x_j)$ as $k \to \infty$ for all $j \in \mathbb{N}_n$. Since the loss function $\mathcal{L}$ is continuous about the first variable, we have that

$$\lim_{k\to\infty}\mathcal{E}_{\mathbf{z}}(f_k) = \mathcal{E}_{\mathbf{z}}(f_0).$$

Also, choosing $g = f_0$ in (61) yields that

$$\lim_{k\to\infty}[f_k, f_0]_{\mathcal{B}} = \|f_0\|_{\mathcal{B}}^2.$$

By the above two equations, for each $\varepsilon > 0$ there exists some N such that for $k \geq N$

$$\mathcal{E}_{\mathbf{z}}(f_0) \leq \mathcal{E}_{\mathbf{z}}(f_k) + \varepsilon$$

and

$$\|f_0\|_{\mathcal{B}}^2 \leq \varepsilon\|f_0\|_{\mathcal{B}}^2 + |[f_k, f_0]_{\mathcal{B}}| \leq \varepsilon\|f_0\|_{\mathcal{B}}^2 + \|f_k\|_{\mathcal{B}}\|f_0\|_{\mathcal{B}}, \text{ if } f_0 \neq 0.$$

If $f_0 = 0$, then trivially we have

$$\|f_0\|_{\mathcal{B}} \leq \varepsilon\|f_0\|_{\mathcal{B}} + \|f_k\|_{\mathcal{B}}.$$

Combining the above three equations, we get for $k \geq N$ that

$$\mathcal{E}_{\mathbf{z},\mu}(f_0) \leq \frac{1}{(1-\varepsilon)^2}\mathcal{E}_{\mathbf{z},\mu}(f_k) + \varepsilon.$$

By (60) and the arbitrariness of ε, we conclude that f_0 is a minimizer of (59).

Since $\mathcal{L}$ is convex with respect to its first variable and $\|\cdot\|_{\mathcal{B}}^2$ is strictly convex, $\mathcal{E}_{\mathbf{z},\mu}$ is strictly convex on $\mathcal{B}$. This implies the uniqueness of the minimizer. ■

In the rest of this subsection, we shall consider the regularization network in $\mathcal{B}$, that is, the loss function $\mathcal{L}$ is specified as

$$\mathcal{L}(a,b) = |a-b|^2, \ \ a,b \in \mathbb{C}.$$

It is continuous and convex with respect to its first variable. Therefore, by Theorem 20, there is a unique minimizer for the regularization network:

$$\min_{f\in\mathcal{B}} \sum_{j\in\mathbb{N}_n} |f(x_j) - y_j|^2 + \mu\|f\|_{\mathcal{B}}^2. \tag{62}$$

We next consider solving the minimizer. To this end, we need the notion of strict convexity of a normed vector space. We call a normed vector space V **strictly convex** if whenever $\|x+y\|_V = \|x\|_V + \|y\|_V$ for some $x, y \neq 0$, there must hold $y = \lambda x$ for some $\lambda > 0$. Note that a uniformly convex normed vector space is automatically strictly convex. The following result was observed in Giles (1967).

Lemma 21 *An s.i.p. space V is strictly convex if and only if whenever $[x,y]_V = \|x\|_V\|y\|_V$ for some $x, y \neq 0$ there holds $y = \lambda x$ for some $\lambda > 0$.*

A technical result about s.i.p. spaces is required for solving the minimizer.

Lemma 22 *Let V be a strictly convex s.i.p. space. Then for all $u, v \in V$*

$$\|u+v\|_V^2 - \|u\|_V^2 - 2\,\mathrm{Re}\,[v,u]_V \geq 0.$$

Proof Assume that there exist $u, v \in V$ such that

$$\|u+v\|_V^2 - \|u\|_V^2 - 2\,\mathrm{Re}\,[v,u]_V < 0.$$

Then we have $u, v \neq 0$. We proceed by the above inequality and properties of semi-inner-products that

$$\begin{aligned} \|u\|_V^2 &= [u+v-v,u]_V = [u+v,u]_V - [v,u]_V \\ &= \mathrm{Re}\,[u+v,u]_V - \mathrm{Re}\,[v,u]_V \leq |[u+v,u]_V| - \mathrm{Re}\,[v,u]_V \\ &\leq \|u+v\|_V\|u\|_V - \frac{\|u+v\|_V^2 - \|u\|_V^2}{2}. \end{aligned}$$

The last inequality above can be simplified as

$$\|u+v\|_V^2 + \|u\|_V^2 \leq 2\|u+v\|_V\|u\|_V.$$

Thus, we must have

$$\|u+v\|_V = \|u\|_V \text{ and } [u+v,u]_V = \|u+v\|_V\|u\|_V.$$

Applying the strict convexity of V and Lemma 21, we obtain that $u+v=u$, namely, $v=0$. This is impossible. ■

Theorem 23 *(Representer Theorem) Let f_0 be the minimizer of (62). Then there exists some $\mathbf{c} \in \mathbb{C}^n$ such that*

$$f_0^* = \sum_{j\in\mathbb{N}_n} c_j G(\cdot,x_j). \tag{63}$$

If $G_{\mathbf{x}}$ is linearly independent then the right hand side of the above equation is the minimizer if and only if

$$\mu\overline{c_k} + \left[G(\cdot,x_k), \sum_{j\in\mathbb{N}_n} c_j G(\cdot,x_j)\right]_{\mathcal{B}^*} = y_k, \ \ k\in\mathbb{N}_n. \tag{64}$$

Proof Let $g\in\mathcal{B}$. Define the function $\phi:\mathbb{R}\to\mathbb{R}$ by $\phi(t) := \mathcal{E}_{\mathbf{z},\mu}(f_0+tg)$, $t\in\mathbb{R}$. We compute by Lemma 7 that

$$\phi'(t) = 2\,\mathrm{Re}\sum_{j\in\mathbb{N}_n} g(x_j)(\overline{f_0(x_j)-y_j} + t\overline{g(x_j)}) + 2\mu\,\mathrm{Re}\,[g,f_0+tg]_{\mathcal{B}}.$$

Since f_0 is the minimizer, $t=0$ is the minimum point of ϕ. Hence $\phi'(0)=0$, that is,

$$\sum_{j\in\mathbb{N}_n} g(x_j)\overline{f_0(x_j)-y_j} + \mu[g,f_0]_{\mathcal{B}} = 0, \ \text{ for all } g\in\mathcal{B}.$$

The above equation can be rewritten as

$$\sum_{j\in\mathbb{N}_n} [\overline{f_0(x_j)-y_j}G(\cdot,x_j),g^*]_{\mathcal{B}^*} + \mu[f_0^*,g^*]_{\mathcal{B}^*} = 0, \ \text{ for all } g\in\mathcal{B},$$

which is equivalent to

$$\mu f_0^* = \sum_{j\in\mathbb{N}_n} \overline{y_j - f_0(x_j)}G(\cdot,x_j). \tag{65}$$

Therefore, f_0^* has the form (63).

If $G_{\mathbf{x}}$ is linearly independent then $f_0^* = \sum_{j\in\mathbb{N}_n} c_j G(\cdot,x_j)$ satisfies (65) if and only if (64) holds. Thus, it remains to show that condition (65) is enough to ensure that f_0 is the minimizer. To this end, we check that (65) leads to

$$\mathcal{E}_{\mathbf{z},\mu}(f_0+g) - \mathcal{E}_{\mathbf{z},\mu}(f_0) = \mu\|f_0+g\|_{\mathcal{B}}^2 - \mu\|f_0\|_{\mathcal{B}}^2 - 2\mu\,\mathrm{Re}\,[g,f_0]_{\mathcal{B}} + \sum_{j\in\mathbb{N}_n}|g(x_j)|^2,$$

which by Lemma 22 is nonnegative. The proof is complete. ■

By Theorem 23, if $G_{\mathbf{x}}$ is linearly independent then the minimizer of (62) can be obtained by solving (64), which has a unique solution in this case. Using the feature map, the system (64) has the following form

$$\mu\overline{c_k} + \left[\Phi^*(x_k), \sum_{j\in\mathbb{N}_n} c_j\Phi^*(x_j)\right]_{\mathcal{B}^*} = y_k, \ \ k\in\mathbb{N}_n.$$

As remarked before, this is in general nonlinear about $\mathbf{c}$.

5.3 Support Vector Machines

In this subsection, we assume that all the spaces are over the field $\mathbb{R}$ of real numbers, and consider learning a classifier from the data $\mathbf{z} := \{(x_j, y_j) : j \in \mathbb{N}_n\} \subseteq X \times \{-1, 1\}$. We shall establish for this task two learning algorithms in RKBS whose RKHS versions are well-known (Evgeniou et al., 2000; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004; Vapnik, 1998; Wahba, 1999).

5.3.1 SOFT MARGIN HYPERPLANE CLASSIFICATION

We first focus on the soft margin hyperplane classification by studying

$$\inf\left\{\frac{1}{2}\|w\|_{\mathcal{W}}^2 + C\|\xi\|_{\ell^1(\mathbb{N}_n)} : w \in \mathcal{W},\ \xi := (\xi_j : j \in \mathbb{N}_n) \in \mathbb{R}_+^n,\ b \in \mathbb{R}\right\} \tag{66}$$

subject to

$$y_j([\Phi(x_j), w]_{\mathcal{W}} + b) \geq 1 - \xi_j, \quad j \in \mathbb{N}_n.$$

Here, C is a fixed positive constant controlling the tradeoff between margin maximization and training error minimization. If a minimizer $(w_0, \xi_0, b_0) \in \mathcal{W} \times \mathbb{R}_+^n \times \mathbb{R}$ exists, the classifier is taken as $\operatorname{sgn}([\Phi(\cdot), w_0]_{\mathcal{W}} + b_0)$.

Recall by Theorem 10 that functions in $\mathcal{B}^*$ are of the form

$$f^* = [\Phi(\cdot), w]_{\mathcal{W}}, \quad w \in \mathcal{W} \tag{67}$$

and

$$\|f^*\|_{\mathcal{B}^*} = \|w\|_{\mathcal{W}}.$$

Introduce the loss function

$$\mathcal{L}_b(a, y) := \max\{1 - ay - by, 0\}, \quad (a, y) \in \mathbb{R} \times \{-1, 1\}, \quad b \in \mathbb{R},$$

and the error functional on $\mathcal{B}^*$,

$$\mathcal{E}_{b,\mathbf{z},\mu}(f^*) := \mu\|f^*\|_{\mathcal{B}^*}^2 + \sum_{j \in \mathbb{N}_n} \mathcal{L}_b(f^*(x_j), y_j), \quad f^* \in \mathcal{B}^*$$

where $\mu := 1/(2C)$. Then we observe that (66) can be equivalently rewritten as

$$\inf\{\mathcal{E}_{b,\mathbf{z},\mu}(f^*) : f^* \in \mathcal{B}^*, \quad b \in \mathbb{R}\}. \tag{68}$$

When $b = 0$, (68) is also called the support vector machine classification (Wahba, 1999).

If a minimizer $(f_0^*, b_0) \in \mathcal{B}^* \times \mathbb{R}$ for (68) exists then by (67), the classifier followed from (66) will be taken as $\operatorname{sgn}(f_0^* + b_0)$. It can be verified that for every $b \in \mathbb{R}$, $\mathcal{L}_b$ is convex and continuous with respect to its first variable. This enables us to prove the existence of minimizers for (68) based on Theorem 20.

Proposition 24 *Suppose that $\{y_j : j \in \mathbb{N}_n\} = \{-1, 1\}$. Then there exists a minimizer $(f_0^*, b_0) \in \mathcal{B}^* \times \mathbb{R}$ for (68). Moreover, the first component f_0^* of the minimizer is unique.*

Proof The uniqueness of f_0^* follows from the fact that $\mathcal{E}_{b,\mathbf{z},\mu}$ is strictly convex with respect to its first variable. It remains to deal with the existence. Let e be the infimum (68). Then there exists a sequence $(f_k^*, b_k^*) \in \mathcal{B}^* \times \mathbb{R}, k \in \mathbb{N}$ such that

$$\lim_{k\to\infty} \mathcal{E}_{b_k,\mathbf{z},\mu}(f_k^*) = e. \tag{69}$$

Since $\{y_j : j \in \mathbb{N}_n\} = \{-1,1\}$, $\{b_k : k \in \mathbb{N}\}$ is bounded on $\mathbb{R}$. We may hence assume that b_k converges to some $b_0 \in \mathbb{R}$. By Theorem 20, $\min\{\mathcal{E}_{b_0,\mathbf{z},\mu}(f^*) : f^* \in \mathcal{B}^*\}$ has a unique minimizer f_0^*. The last fact we shall need is the simple observation that

$$\max\{1-ay-by,0\} - \max\{1-ay-b'y,0\} \le |b-b'| \text{ for all } a,b,b' \in \mathbb{R} \text{ and } y \in \{-1,1\}.$$

Thus, we get for all $k \in \mathbb{N}$ that

$$\mathcal{E}_{b_0,\mathbf{z},\mu}(f_0^*) \le \mathcal{E}_{b_0,\mathbf{z},\mu}(f_k^*) \le \mathcal{E}_{b_k,\mathbf{z},\mu}(f_k^*) + n|b_0 - b_k|,$$

which together with (69) and $\lim_{k\to\infty} b_k = b_0$ implies that (f_0^*, b_0) is a minimizer for (68). ■

Since the soft margin hyperplane classification (66) is equivalent to (68), we obtain by Proposition 24 that it has a minimizer $(w_0, \xi_0, b_0) \in \mathcal{W} \times \mathbb{R}_+^n \times \mathbb{R}$, where the first component w_0 is unique.

We shall prove a representer theorem for f_0^* using the following celebrated geometric consequence of the Hahn-Banach theorem (see, Conway, 1990, page 111).

Lemma 25 *Let A be a closed convex subset of a normed vector space V and b a point in V that is not contained in A. Then there exist $T \in V^*$ and $\alpha \in \mathbb{R}$ such that*

$$T(b) < \alpha < T(a), \text{ for all } a \in A.$$

Theorem 26 *(Representer Theorem) Let f_0^* be the minimizer of (68). Then f_0 lies in the closed convex cone $\operatorname{co} G_\mathbf{z}$ spanned by $G_\mathbf{z} := \{y_j G(x_j, \cdot) : j \in \mathbb{N}_n\}$, that is, there exist $\lambda_j \ge 0$ such that*

$$f_0 = \sum_{j\in\mathbb{N}_n} \lambda_j y_j G(x_j, \cdot). \tag{70}$$

Consequently, the minimizer w_0 of (66) belongs to the closed convex cone spanned by $y_j\Phi(x_j)$, $j \in \mathbb{N}_n$.

Proof Assume that $f_0 \notin \operatorname{co} G_\mathbf{z}$. By Lemmas 25 and 8, there exists $g \in \mathcal{B}$ and $\alpha \in \mathbb{R}$ such that for all $\lambda \ge 0$

$$[f_0, g]_\mathcal{B} < \alpha < [\lambda y_j G(x_j, \cdot), g]_\mathcal{B} = \lambda y_j g^*(x_j), \quad j \in \mathbb{N}_n.$$

Choosing $\lambda = 0$ above yields that $\alpha < 0$. That $\lambda y_j g^*(x_j) > \alpha$ for all $\lambda \ge 0$ implies $y_j g^*(x_j) \ge 0$, $j \in \mathbb{N}_n$. We hence obtain that

$$[f_0, g]_\mathcal{B} < 0 \le y_j g^*(x_j), \quad j \in \mathbb{N}_n.$$

We choose $f^* = f_0^* + tg^*, t > 0$. First, observe from $y_j g^*(x_j) \ge 0$ that

$$1 - y_j f^*(x_j) \le 1 - y_j f_0^*(x_j), \quad j \in \mathbb{N}_n. \tag{71}$$

By Lemma 7,

$$\lim_{t\to 0^+} \frac{\|f_0^* + tg^*\|_{\mathcal{B}^*}^2 - \|f_0^*\|_{\mathcal{B}^*}^2}{t} = 2[g^*, f_0^*]_{\mathcal{B}^*} = 2[f_0, g]_{\mathcal{B}} < 0.$$

Therefore, there exists $t > 0$ such that $\|f^*\|_{\mathcal{B}^*}^2 < \|f_0^*\|_{\mathcal{B}^*}^2$. This combined with (71) implies that $\mathcal{E}_{b,\mathbf{z},\mu}(f^*) < \mathcal{E}_{b,\mathbf{z},\mu}(f_0^*)$ for every $b \in \mathbb{R}$, contradicting the hypothesis that f_0^* is the minimizer. ■

To solve (68), by Theorem 26 one substitutes equation (70) into (68) to obtain a convex optimization problem about λ_j subject to the constraint that $\lambda_j \geq 0$, $j \in \mathbb{N}_n$.

5.3.2 HARD MARGIN HYPERPLANE CLASSIFICATION

Consider in the feature space $\mathcal{W}$ the following hard margin classification problem

$$\inf\{\|w\|_{\mathcal{W}} : w \in \mathcal{W},\ b \in \mathbb{R}\} \tag{72}$$

subject to

$$y_j([\Phi(x_j), w]_{\mathcal{W}} + b) \geq 1, \quad j \in \mathbb{N}_n.$$

Provided that the minimizer $(w_0, b_0) \in \mathcal{W} \times \mathbb{R}$ exists, the classifier is $\operatorname{sgn}([\Phi(\cdot), w_0]_{\mathcal{W}} + b_0)$.

Hard margin classification in s.i.p. spaces was discussed in Der and Lee (2007). Applying the results in our setting tells that if b is fixed then (72) has a unique minimizer w_0 and $w_0 \in \operatorname{co}\{y_j\Phi(x_j) : j \in \mathbb{N}_n\}$. As a corollary of Proposition 24 and Theorem 26, we obtain here that if $\{y_j : j \in \mathbb{N}_n\} = \{-1, 1\}$ then (72) has a minimizer (w_0, b_0), where w_0 is unique and belongs to the set $\operatorname{co}\{y_j\Phi(x_j) : j \in \mathbb{N}_n\}$.

We draw the conclusion that the support vector machine classifications in this subsection all reduce to a convex optimization problem.

5.4 Kernel Principal Component Analysis

Kernel principal component analysis (PCA) plays a foundational role in data preprocessing for other learning algorithms. We shall present an extension of kernel PCA for RKBS. To this end, let us briefly review the classical kernel PCA (see, for example, Schölkopf and Smola, 2002; Schölkopf et al., 1998).

Let $\mathbf{x} := \{x_j : j \in \mathbb{N}_n\} \subseteq X$ be a set of inputs. We denote by $d(w, V)$ the distance from $w \in \mathcal{W}$ to a closed subspace V of $\mathcal{W}$. Fix $m \in \mathbb{N}$. For each subspace $V \subseteq \mathcal{W}$ with dimension $\dim V = m$, we define the distance from V to $\Phi(\mathbf{x})$ as

$$\mathcal{D}(V, \Phi(\mathbf{x})) := \frac{1}{n} \sum_{j\in\mathbb{N}_n} (d(\Phi(x_j), V))^2.$$

Suppose that $\{u_j : j \in \mathbb{N}_m\}$ is a basis for V. Then for each $v \in \mathcal{W}$ the best approximation v_0 in V of v exists. Assume that $v_0 = \sum_{j\in\mathbb{N}_m} \lambda_j u_j$, $\lambda_j \in \mathbb{C}$, $j \in \mathbb{N}_m$. By Lemma 17, the coefficients λ_j's are uniquely determined by

$$\left[u_k, v - \sum_{j\in\mathbb{N}_m} \lambda_j u_j\right]_{\mathcal{W}} = 0, \quad k \in \mathbb{N}_m. \tag{73}$$

In the case when $\mathcal{W}$ is a Hilbert space, the system (73) of equations resulting from best approximation is linear about λ_j's. This enables us to construct a unique m-dimensional subspace $V_0 \subseteq \mathcal{W}$ such that

$$\mathcal{D}(V_0, \Phi(\mathbf{x})) = \min\{\mathcal{D}(V, \Phi(\mathbf{x})) : V \subseteq \mathcal{W} \text{ subspace, } \dim V = m\}. \tag{74}$$

Let T be the compact operator on $\mathcal{W}$ determined by

$$(Tu, v)_{\mathcal{W}} = \frac{1}{n} \sum_{j \in \mathbb{N}_n} (u, \Phi(x_j))_{\mathcal{W}} (\Phi(x_j), v)_{\mathcal{W}}, \quad u, v \in \mathcal{W}. \tag{75}$$

We let v_j, $j \in \mathbb{N}_m$ be the unit eigenvectors of T corresponding to its first m largest eigenvalues. Then v_j's form an orthonormal basis for V_0 and are called the *principal components* of $\Phi(\mathbf{x})$. For each $x \in X$, $((\Phi(x), v_j)_{\mathcal{W}} : j \in \mathbb{N}_m) \in \mathbb{C}^m$ is its new *feature*. Therefore, kernel PCA amounts to selecting the new feature map from X to $\mathbb{C}^m$. The dimension m is usually chosen to be much smaller than the original dimension of $\mathcal{W}$. Moreover, by (74), the new features of $\mathbf{x}$ are expected to become sparser under this mapping.

The analysis of PCA in Hilbert spaces breaks in s.i.p. spaces where (73) is nonlinear. To tackle this problem, we suggest using a class of linear functionals to measure the distance between two elements in $\mathcal{W}$. Specifically, we choose $B \subseteq \mathcal{W}^*$ and set for all $u, v \in \mathcal{W}$

$$d_B(u, v) := \left(\sum_{b \in B} |(u - v, b)_{\mathcal{W}}|^2 \right)^{1/2}.$$

The idea is that if $d_B(u, v)$ is small for a carefully chosen set B of linear functionals then $\|u - v\|_{\mathcal{W}}$ should be small, and vice versa. In particular, if $\mathcal{W}$ is a Hilbert space and B is an orthonormal basis for $\mathcal{W}$ then $d_B(u, v) = \|u - v\|_{\mathcal{W}}$. From the practical consideration, we shall use what we have at hand, that is, $\Phi(\mathbf{x})$. Thus, we define for each $u, v \in \mathcal{W}$

$$d_{\Phi(\mathbf{x})}(u, v) := \left(\sum_{j \in \mathbb{N}_n} \left| [u - v, \Phi(x_j)]_{\mathcal{W}} \right|^2 \right)^{1/2}.$$

This choice of distance is equivalent to mapping X into $\mathbb{C}^n$ by

$$\widetilde{\Phi}(x) := ([\Phi(x), \Phi(x_j)]_{\mathcal{W}} : j \in \mathbb{N}_n), \quad x \in X.$$

Consequently, new features of elements in X will be obtained by applying the classical PCA to $\widetilde{\Phi}(x_j)$, $j \in \mathbb{N}_n$ in the Hilbert space $\mathbb{C}^n$.

In our method the operator T defined by (75) on $\mathbb{C}^n$ is of the form

$$Tu = \frac{1}{n} \sum_{j \in \mathbb{N}_n} (\widetilde{\Phi}(x_j)^* u) \widetilde{\Phi}(x_j), \quad u \in \mathbb{C}^n,$$

where $\widetilde{\Phi}(x_j)^*$ is the conjugate transpose of $\widetilde{\Phi}(x_j)$. One can see that T has the matrix representation $Tu = M_{\mathbf{x}} u$, $u \in \mathbb{C}^n$, where

$$M_{\mathbf{x}} := \frac{1}{n} (G[\mathbf{x}]^* G[\mathbf{x}])^T.$$

Let $\lambda_k, k \in \mathbb{N}_n$, be the eigenvalues of $M_{\mathbf{x}}$ arranged in nondecreasing order. We find for each $k \in \mathbb{N}_m$ the unit eigenvector $\alpha^k := (\alpha_j^k : j \in \mathbb{N}_n) \in \mathbb{C}^n$ corresponding to λ_k, that is,

$$M_{\mathbf{x}}\alpha^k = \lambda_k \alpha^k.$$

Vectors $\alpha^k, k \in \mathbb{N}_m$, form an orthonormal sequence. The new feature for $x \in X$ is hence

$$((\widetilde{\Phi}(x), \alpha^k)_{\mathbb{C}^n} : k \in \mathbb{N}_m) \in \mathbb{C}^m.$$

We compute explicitly that

$$(\widetilde{\Phi}(x), \alpha^k)_{\mathbb{C}^n} = \sum_{j \in \mathbb{N}_n} \overline{\alpha_j^k} G(x, x_j), \quad k \in \mathbb{N}_m.$$

We remark that unlike the previous three learning algorithms, the kernel PCA presented here only makes use of the kernel G and is independent of the semi-inner-product on $\mathcal{W}$. The *kernel trick* can hence be applied to this algorithm.

6. Conclusion

We have introduced the notion of reproducing kernel Banach spaces and generalized the correspondence between an RKHS and its reproducing kernel to the setting of RKBS. S.i.p. RKBS were specially treated by making use of semi-inner-products and the duality mapping. A semi-inner-product shares many useful properties of an inner product. These properties and the general theory of semi-inner-products make it possible to develop many learning algorithms in RKBS. As illustration, we discussed in the RKBS setting the minimal norm interpolation, regularization network, support vector machines, and kernel PCA. Various representer theorems were established.

This work attempts to provide an appropriate mathematical foundation of kernel methods for learning in Banach spaces. Many theoretical and practical issues are left for future research. An immediate challenge is to construct a class of useful RKBS and the corresponding reproducing kernels. By the classical theory of RKHS, a function K is a reproducing kernel if and only if the finite matrix (1) is always hermitian and positive semi-definite. This function property characterization brings great convenience to the construction of positive definite kernels. Thus, we ask what characteristics a function must possess so that it is a reproducing kernel for some RKBS. Properties of RKBS and their reproducing kernels also deserve a systematic study. For the applications, we have seen that minimum norm interpolation and regularization network reduce to systems of nonlinear equations. Dealing with the nonlinearity requires algorithms specially designed for the underlying s.i.p. space. On the other hand, support vector machines can be reformulated into certain convex optimization problems. Finally, section 5.4 only provides a possible implementation of kernel PCA for RKBS. We are interested in further careful analysis and efficient algorithms for these problems. We shall return to these issues in future work.

Acknowledgments

Yuesheng Xu was supported in part by the US National Science Foundation under grant DMS-0712827. Jun Zhang was supported in part by the US National Science Foundation under grant

0631541. This research was accomplished when the last author (J.Z.) was on leave from the University of Michigan to AFOSR under an IPA assignment.

Send all correspondence to Yuesheng Xu.

Appendix A.

In this appendix, we provide proofs of two results stated in the previous sections of this paper. The first one is about the minimization problem (5) in the introduction.

Proposition 27 *If $\varphi : \mathbb{R}^d \to [0,+\infty)$ is strictly concave and $\mu > 0$, then every minimizer of*

$$\min\{\varphi(x) + \mu\|x\|_{\ell^1} : x \in \mathbb{R}^d\} \tag{76}$$

has at most one nonzero element.

Proof Assume to the contrary that $x_0 \in \mathbb{R}^d$ is a minimizer of (76) with more than one nonzero elements. Then x_0 is not an extreme points of the sphere $\{x \in \mathbb{R}^d : \|x\|_{\ell^1} = \|x_0\|_{\ell^1}\}$. In other words, there exist two distinct vectors $x_1, x_2 \in \mathbb{R}^d$ and some $\lambda \in (0,1)$ such that

$$x_0 = \lambda x_1 + (1-\lambda)x_2 \text{ and } \|x_1\|_{\ell^1} = \|x_2\|_{\ell^1} = \|x_0\|_{\ell^1}.$$

By the strict concavity of φ, we get that

$$\begin{aligned} \varphi(x_0) + \mu\|x_0\|_{\ell^1} &> \lambda\varphi(x_1) + (1-\lambda)\varphi(x_2) + \mu\|x_0\|_{\ell^1} \\ &= \lambda(\varphi(x_1) + \mu\|x_1\|_{\ell^1}) + (1-\lambda)(\varphi(x_2) + \mu\|x_2\|_{\ell^1}). \end{aligned}$$

Therefore, we must have either

$$\varphi(x_0) + \mu\|x_0\|_{\ell^1} > \varphi(x_1) + \mu\|x_1\|_{\ell^1}$$

or

$$\varphi(x_0) + \mu\|x_0\|_{\ell^1} > \varphi(x_2) + \mu\|x_2\|_{\ell^1}.$$

Either case contradicts the hypothesis that x_0 is a minimizer of (76). ■

The second result confirms that (27) indeed defines a semi-inner-product.

Proposition 28 *Let V be a normed vector space over $\mathbb{C}$. If for all $x, y \in V \setminus \{0\}$ the limit $\lim_{t\in\mathbb{R}, t\to 0} \frac{\|x+ty\|_V - \|x\|_V}{t}$ exists then $[\cdot,\cdot]_V : V \times V \to \mathbb{C}$ defined by*

$$[x,y]_V := \|y\|_V \left(\lim_{t\in\mathbb{R}, t\to 0} \frac{\|y+tx\|_V - \|y\|_V}{t} + i \lim_{t\in\mathbb{R}, t\to 0} \frac{\|iy+tx\|_V - \|y\|_V}{t} \right) \text{ if } x, y \neq 0 \tag{77}$$

and $[x,y]_V := 0$ if $x = 0$ or $y = 0$ is a semi-inner-product on V.

Proof First, we obtain for $x \neq 0$ that

$$\begin{aligned}[x,x]_V &= \|x\|_V \left(\lim_{t\in\mathbb{R},t\to 0} \frac{\|(1+t)x\|_V - \|x\|_V}{t} + i \lim_{t\in\mathbb{R},t\to 0} \frac{\|(i+t)x\|_V - \|x\|_V}{t} \right) \\ &= \|x\|_V^2 \left(\lim_{t\in\mathbb{R},t\to 0} \frac{|1+t|-1}{t} + i \lim_{t\in\mathbb{R},t\to 0} \frac{|i+t|-1}{t} \right) \\ &= \|x\|_V^2 (1+0) = \|x\|_V^2 > 0. \end{aligned} \tag{78}$$

We then deal with the remaining three conditions of a semi-inner-product. Clearly, they are true if one of the arguments involved is the zero vector. Let $x,y,z \in V \setminus \{0\}$. We start with the estimate:

$$\begin{aligned}\operatorname{Re}[x+y,z]_V &= \|z\|_V \lim_{t\in\mathbb{R},t\to 0^+} \frac{\|z+tx+ty\|_V - \|z\|_V}{t} \\ &\leq \|z\|_V \lim_{t\in\mathbb{R},t\to 0^+} \frac{\|\frac{z}{2}+tx\|_V + \|\frac{z}{2}+ty\|_V - \|z\|_V}{t} \\ &= \|z\|_V \left(\lim_{t\in\mathbb{R},t\to 0^+} \frac{\|\frac{z}{2}+tx\|_V - \|\frac{z}{2}\|_V}{t} + \lim_{t\in\mathbb{R},t\to 0^+} \frac{\|\frac{z}{2}+ty\|_V - \|\frac{z}{2}\|_V}{t} \right) \\ &= \|z\|_V \left(\lim_{t\in\mathbb{R},t\to 0^+} \frac{\|z+2tx\|_V - \|z\|_V}{2t} + \lim_{t\in\mathbb{R},t\to 0^+} \frac{\|z+2ty\|_V - \|z\|_V}{2t} \right). \end{aligned}$$

The above equation implies that

$$\operatorname{Re}[x+y,z]_V \leq \operatorname{Re}[x,z]_V + \operatorname{Re}[y,z]_V. \tag{79}$$

It can be easily verified that $[-x,y]_V = -[x,y]_V$. Replacing y with $x+y$, and x with $-x$ in the above equation yields that

$$\operatorname{Re}[y,z]_V \leq \operatorname{Re}[-x,z]_V + \operatorname{Re}[x+y,z]_V = -\operatorname{Re}[x,z]_V + \operatorname{Re}[x+y,z]_V. \tag{80}$$

Combining (79) and (80), we get that $\operatorname{Re}[x+y,z]_V = \operatorname{Re}[x,z]_V + \operatorname{Re}[y,z]_V$. Similar arguments lead to that $\operatorname{Im}[x+y,z]_V = \operatorname{Im}[x,z]_V + \operatorname{Im}[y,z]_V$. Therefore,

$$[x+y,z]_V = [x,z]_V + [y,z]_V. \tag{81}$$

Next we see for all $\lambda \in \mathbb{R} \setminus \{0\}$ that

$$[\lambda x,y]_V = \|y\|_V \lim_{t\in\mathbb{R},t\to 0} \frac{\|y+t\lambda x\|_V - \|y\|_V}{t} = \lambda \|y\|_V \lim_{t\in\mathbb{R},t\to 0} \frac{\|y+t\lambda x\|_V - \|y\|_V}{\lambda t} = \lambda [x,y]_V.$$

It is also clear from the definition (77) that $[ix,y]_V = i[x,y]_V$. We derive from these two facts and (81) for every $\lambda = \alpha + i\beta$, $\alpha,\beta \in \mathbb{R}$ that

$$\begin{aligned}[\lambda x,y]_V &= [\alpha x + i\beta x, y]_V = [\alpha x,y]_V + [i\beta x,y]_V = \alpha[x,y]_V + i[\beta x,y]_V \\ &= \alpha[x,y]_V + i\beta[x,y]_V = (\alpha+i\beta)[x,y]_V = \lambda[x,y]_V. \end{aligned} \tag{82}$$

We then proceed for $\lambda \in \mathbb{C} \setminus \{0\}$ by (82) that

$$\begin{aligned}[x,\lambda y]_V &= \|\lambda y\|_V \lim_{t\in\mathbb{R},t\to 0} \frac{\|\lambda y + tx\|_V - \|\lambda y\|_V}{t} = \|\lambda y\|_V |\lambda| \lim_{t\in\mathbb{R},t\to 0} \frac{\|y + t\frac{x}{\lambda}\|_V - \|y\|_V}{t} \\ &= |\lambda|^2 [\frac{x}{\lambda}, y]_V = \frac{|\lambda|^2}{\lambda} [x,y]_V = \overline{\lambda}[x,y]_V. \end{aligned} \tag{83}$$

Finally, we find some $\lambda \in \mathbb{C}$ such that $|\lambda| = 1$ and $\lambda[x,y]_V = |[x,y]_V|$, and then obtain by (82) and (77) that

$$\begin{aligned} |[x,y]_V| &= \lambda[x,y]_V = [\lambda x, y]_V = \|y\|_V \lim_{t\in\mathbb{R}, t\to 0^+} \frac{\|y + t\lambda x\|_V - \|y\|_V}{t} \\ &\leq \|y\|_V \lim_{t\in\mathbb{R}, t\to 0^+} \frac{\|y\|_V + t\|\lambda x\|_V - \|y\|_V}{t} = \|y\|_V \|\lambda x\|_V = \|x\|_V \|y\|_V. \end{aligned}$$

By (78), the above inequality has the equivalent form

$$|[x,y]_V| \leq [x,x]_V^{1/2} [y,y]_V^{1/2}. \tag{84}$$

Combining Equations (78), (81), (82), (83), and (84) proves the proposition. ■

References

A. Argyriou, C. A. Micchelli and M. Pontil. When is there a representer theorem? Vector versus matrix regularizers. arXiv:0809.1590v1.

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68: 337–404, 1950.

K. P. Bennett and E. J. Bredensteiner. Duality and geometry in SVM classifiers. In *Proceeding of the Seventeenth International Conference on Machine Learning*, pages 57–64, Morgan Kaufmann, San Francisco, 2000.

S. Canu, X. Mary and A. Rakotomamonjy. Functional learning through kernel. In *Advances in Learning Theory: Methods, Models and Applications*, pages 89–110, NATO Science Series III: Computer and Systems Sciences, Volume 190, IOS Press, Amsterdam, 2003.

A. Caponnetto, C. A. Micchelli, M. Pontil and Y. Ying. Universal multi-task kernels. *Journal of Machine Learning Research*, 9: 1615–1646, 2008.

J. B. Conway. *A Course in Functional Analysis*. 2nd Edition, Springer-Verlag, New York, 1990.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc.*, 39: 1–49, 2002.

D. F. Cudia. On the localization and directionalization of uniform convexity. *Bull. Amer. Math. Soc.*, 69: 265–267, 1963.

R. Der and D. Lee. Large-margin classification in Banach spaces. *JMLR Workshop and Conference Proceedings*, 2: AISTATS: 91–98, 2007.

T. Evgeniou, M. Pontil and T. Poggio. Regularization networks and support vector machines. *Adv. Comput. Math.*, 13: 1–50, 2000.

M. J. Fabian, P. Habala, P. Hajek and J. Pelant. *Functional Analysis and Infinite-Dimensional Geometry*. Springer, New York, 2001.

C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2: 213–242, 2001.

J. R. Giles. Classes of semi-inner-product spaces. *Trans. Amer. Math. Soc.*, 129: 436–446, 1967.

M. Hein, O. Bousquet and B. Schölkopf. Maximal margin classification for metric spaces. *J. Comput. System Sci.*, 71: 333–359, 2005.

V. I. Istrăţescu. *Strict Convexity and Complex Strict Convexity: Theory and Applications*. Lecture Notes in Pure and Applied Mathematics 89, Marcel Dekker, New York, 1984.

D. Kimber and P. M. Long. On-line learning of smooth functions of a single variable. *Theoret. Comput. Sci.*, 148: 141–156, 1995.

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, 33: 82–95, 1971.

G. Lumer. Semi-inner-product spaces. *Trans. Amer. Math. Soc.*, 100: 29–43, 1961.

J. Mercer. Functions of positive and negative type and their connection with the theorey of integral equations. *Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.*, 209: 415–446, 1909.

C. A. Micchelli and M. Pontil. A function representation for learning in Banach spaces. In *Learning Theory*, pages 255–269, Lecture Notes in Computer Science 3120, Springer, Berlin, 2004.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Comput.*, 17: 177–204, 2005.

C. A. Micchelli and M. Pontil. Feature space perspectives for learning the kernel. *Machine Learning*, 66: 297–319, 2007.

C. A. Micchelli, Y. Xu and P. Ye. Cucker Smale learning theory in Besov spaces. In *Advances in Learning Theory: Methods, Models and Applications*, pages 47–68, IOS Press, Amsterdam, The Netherlands, 2003.

C. A. Micchelli, Y. Xu and H. Zhang. Universal kernels. *Journal of Machine Learning Research*, 7: 2651–2667, 2006.

C. A. Micchelli, Y. Xu and H. Zhang. Optimal learning of bandlimited functions from localized sampling. *J. Complexity*, 25: 85–114, 2009.

W. Rudin. *Real and Complex Analysis*. 3rd Edition, McGraw-Hill, New York, 1987.

S. Saitoh. *Integral Transforms, Reproducing Kernels and Their Applications*. Pitman Research Notes in Mathematics Series 369, Longman, Harlow, 1997.

B. Schölkopf, R. Herbrich and A. J. Smola. A generalized representer theorem. In *Proceeding of the 14th Annual Conference on Computational Learning Theory and the 5th European Conference on Computational Learning Theory*, pages 416–426, Springer-Verlag, London, UK, 2001.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, Mass, 2002.

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

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, 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.

J. A. Tropp. Just relax: convex programming methods for identifying sparse signals in noise. *IEEE Trans. Inform. Theory*, 52: 1030–1051, 2006.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

U. von Luxburg and O. Bousquet. Distance-based classification with Lipschitz functions. *Journal of Machine Learning Research*, 5: 669–695, 2004.

G. Wahba. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In *Advances in Kernel Methods–Support Vector Learning*, pages 69–86, MIT Press, Cambridge, Mass, 1999.

Y. Xu and H. Zhang. Refinable kernels. *Journal of Machine Learning Research*, 8: 2083–2120, 2007.

Y. Xu and H. Zhang. Refinement of reproducing kernels. *Journal of Machine Learning Research*, 10: 107–140, 2009.

T. Zhang. On the dual formulation of regularized linear systems with convex risks. *Machine Learning*, 46: 91–129, 2002.

D. Zhou, B. Xiao, H. Zhou and R. Dai. Global geometry of SVM classifiers. *Technical Report 30-5-02*, Institute of Automation, Chinese Academy of Sciences, 2002.

Journal of Machine Learning Research 10 (2009) 2777-2836 Submitted 2/09; Revised 8/09; Published 12/09

Cautious Collective Classification

Luke K. McDowell LMCDOWEL@USNA.EDU
Department of Computer Science
U.S. Naval Academy
Annapolis, MD 21402, USA

Kalyan Moy Gupta KALYAN.GUPTA@KNEXUSRESEARCH.COM
Knexus Research Corporation
Springfield, VA 22153, USA

David W. Aha DAVID.AHA@NRL.NAVY.MIL
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory (Code 5514)
Washington, DC 20375, USA

Editor: Michael Collins

Abstract

Many collective classification (CC) algorithms have been shown to increase accuracy when instances are interrelated. However, CC algorithms must be carefully applied because their use of estimated labels can in some cases decrease accuracy. In this article, we show that managing this label uncertainty through *cautious* algorithmic behavior is essential to achieving maximal, robust performance. First, we describe *cautious inference* and explain how four well-known families of CC algorithms can be parameterized to use varying degrees of such caution. Second, we introduce *cautious learning* and show how it can be used to improve the performance of almost any CC algorithm, with or without cautious inference. We then evaluate cautious inference and learning for the four collective inference families, with three local classifiers and a range of both synthetic and real-world data. We find that cautious learning and cautious inference typically outperform less cautious approaches. In addition, we identify the data characteristics that predict more substantial performance differences. Our results reveal that *the degree of caution used usually has a larger impact on performance than the choice of the underlying inference algorithm*. Together, these results identify the most appropriate CC algorithms to use for particular task characteristics and explain multiple conflicting findings from prior CC research.

Keywords: collective inference, statistical relational learning, approximate probabilistic inference, networked data, cautious inference

1. Introduction

Traditional methods for supervised learning assume that the instances to be classified are independent of each other. However, in many classification tasks, instances can be related. For example, hyperlinked web pages are more likely to have the same class label than unlinked pages. Such autocorrelation (correlation of class labels among interrelated instances) exists in a wide variety of data (Neville and Jensen, 2007; Macskassy and Provost, 2007), including situations where the relationships are implicit (e.g., email messages between two people are likely to share topics).

Collective classification (CC) is a method for jointly classifying related instances. To do so, CC methods employ a *collective inference* algorithm that exploits dependencies between instances (e.g., autocorrelation), enabling CC to often attain higher accuracies than traditional methods when instances are interrelated (Neville and Jensen, 2000; Taskar et al., 2002; Jensen et al., 2004; Sen et al., 2008). Several algorithms have been used for collective inference, including relaxation labeling (Chakrabarti et al., 1998), the iterative classification algorithm (*ICA*) (Lu and Getoor, 2003a), loopy belief propagation (*LBP*) (Taskar et al., 2002), Gibbs sampling (*Gibbs*) (Jensen et al., 2004), and variants of the weighted-vote relational neighbor algorithm (*wvRN*) (Macskassy and Provost, 2007).

During testing, all collective inference algorithms exploit relational features based on uncertain estimation of class labels. This test-time label uncertainty can diminish accuracy due to two related effects. First, an incorrectly predicted label during testing may negatively influence the predictions of its linked neighbors, possibly leading to cascading inference errors (cf., Neville and Jensen, 2008). Second, the training process may learn a poor model for test-time inference, because of the disparity between the training scenario (where labels are known and certain) and the test scenario (where labels are estimated and hence possibly incorrect). As a result, while CC has many potential advantages, in some cases CC's label uncertainty may actually cause accuracy to decrease compared to non-relational approaches (Neville and Jensen, 2007; Sen and Getoor, 2006; Sen et al., 2008).

In this article, we argue that managing this test-time label uncertainty through "cautious" algorithmic behavior is essential to achieving maximal, robust performance. We describe two complementary cautious strategies. Each addresses the fundamental problem of label uncertainty, but separately targets the two manifestations of the problem described above. First, *cautious inference* is an inference process that attends to the uncertainty of its intermediate label predictions. For example, existing algorithms such as *Gibbs* or *LBP* accomplish cautious inference by sampling from or directly reasoning with the estimated label distributions. These techniques are cautious because they prevent less certain label estimates from having substantial influence on subsequent estimations. Alternatively, we show how variants of a simpler algorithm, *ICA*, can perform cautious inference by appropriately favoring more certain information. Second, *cautious learning* refers to a training process that ameliorates the aforementioned train/test disparity. In particular, we introduce PLUL (Parameter Learning for Uncertain Labels), which uses standard cross-validation techniques, but in a way that is new for CC and that leads to significant performance advantages. In particular, PLUL is cautious because it prevents the algorithm from learning a model from the (correctly labeled) training set that overestimates how useful relational features will be when computed with uncertain labels from the test set.

We consider four frequently-studied families of CC algorithms: *ICA*, *Gibbs*, *LBP*, and *wvRN*. For each family, we describe algorithms that use varying degrees of cautious inference and explain how they all (except for the relational-only *wvRN*) can also exploit cautious learning via PLUL. We then evaluate the variants of these four families, with and without PLUL, over a wide range of synthetic and real-world data sets. To broaden the evidence for our results, we evaluate three local classifiers that are used by some of the CC algorithms, and also compare against a non-relational baseline.

While recent CC studies describe complementary results and make some related comparisons, they omit important variations that we consider here (see Section 3). Moreover, the scope and/or methodology of previous studies leaves several important questions unanswered. For instance, *Gibbs* is often regarded as one of the most accurate inference algorithms, and has been shown

to work well for CC (Jensen et al., 2004; Neville and Jensen, 2007). If so, why did Sen et al. (2008) find no significant difference between *Gibbs* and the much less sophisticated *ICA*? Second, we earlier reported that ICA_C (a cautious variant of *ICA*) outperforms both *Gibbs* and *ICA* on three real-world data sets (McDowell et al., 2007a). Why would ICA_C outperform *Gibbs*, and for what data characteristics are ICA_C's gains significant? We answer these questions and more in Section 8.

We hypothesize that *cautious CC algorithms will outperform more aggressive CC approaches when there exists a high probability of an "incorrect relational inference"*, which we define as a prediction error that is due to reasoning with relational features (i.e., an error that does not occur when relational features are removed). Two kinds of data characteristics increase the likelihood of such errors. First, when the data characteristics lead to lower overall classification accuracy (e.g., when the non-relational attributes are not highly predictive), then the computed relational feature values will be less reliable. Second, when a typical relational link is highly predictive (e.g., as occurs when the data exhibits high *relational autocorrelation*), then the potential effect of any incorrect prediction is magnified. As the magnitude of either of these data set characteristics increases, cautious algorithms should outperform more aggressive algorithms by an increasing amount.

Our contributions are as follows. First, we describe cautious inference and how four commonly-used families of existing CC inference algorithms can exhibit more or less caution. Second, we introduce cautious learning and explain how it can help compensate for the train/test disparity that occurs when a CC algorithm uses estimated class labels during testing. Third, we identify the data characteristics for which these cautious techniques should outperform more aggressive approaches, as introduced in the preceding paragraph and discussed in more detail in Section 6. Our experimental results confirm that cautious approaches typically do outperform less cautious variants, and that these effects grow larger when there is a greater probability of incorrect relational inference. Moreover, our results reveal that in most cases *the degree of caution used has a larger impact on performance than the choice of the underlying inference algorithm*. In particular, the cautious algorithms perform very similarly, regardless of whether ICA_C or *Gibbs* or *LBP* is used, although our results also confirm that, for some data characteristics, inference with *LBP* performs comparatively poorly. These results suggest that in many cases the higher computational complexity of *Gibbs* and *LBP* is unnecessary, and that the much faster ICA_C should be used instead. Finally, our results and analysis enable us to answer the previously mentioned questions regarding CC.

The next two sections summarize collective classification and related work. Section 4 then explains why CC needs to be cautious and describes cautious inference and learning in more detail. In Section 5, we describe how caution can be specifically used by the four families of CC inference algorithms. Section 6 then describes our methodology and hypotheses. Section 7 presents our results, which we discuss in Section 8. We conclude in Section 9.

2. Collective Classification: Description and Problem Definition

In this section, we first motivate and define collective classification (CC). We then describe different approaches to CC, different CC tasks, and our assumptions for this article.

2.1 Problem Statement and Example

In many domains, relations exist among instances (e.g., among hyperlinked web pages, social network members, co-cited publications). These relations may be helpful for classification tasks, such

as predicting the topic of a publication or the group membership of a person (Koller et al., 2007). More formally, we consider the following task (based on Macskassy and Provost, 2006):

Definition 1 *(Classification of Graph-based Data) Assume we are given a graph $G = (V, E, X, Y, C)$ where V is a set of nodes, E is set of (possibly directed) edges, each $\vec{x}_i \in X$ is an attribute vector for node $v_i \in V$, each $Y_i \in Y$ is a label variable for v_i, and C is the set of possible labels. Assume further that we are given a set of "known" values Y^K for nodes $V^K \subset V$, so that $Y^K = \{y_i | v_i \in V^K\}$. Then the task is to infer Y^U, the values of Y_i for the remaining nodes with "unknown" values ($V^U = V - V^K$), or a probability distribution over those values.*[1]

For example, consider the task of predicting whether a web page belongs to a professor or a student. Conventional supervised learning approaches ignore the link relations and classify each page using attributes derived from its content (e.g., words present in the page). We refer to this approach as *non-relational classification*. In contrast, a technique for *relational classification* would explicitly use the links to construct additional relational features for classification (e.g., for each page, including as features the words from hyperlinked pages). This additional information can potentially increase classification accuracy, though may sometimes decrease accuracy as well (Chakrabarti et al., 1998). Alternatively, even greater (and usually more reliable) increases can occur when the class *labels* of the linked pages are used instead to derive relevant relational features (Jensen et al., 2004). However, using features based on these labels is challenging, because some or all of the labels are initially unknown, and thus typically must be estimated and then iteratively refined in some way. This process of jointly inferring the labels of interrelated nodes is known as *collective classification* (CC).

Figure 1 summarizes an example execution of a simple CC algorithm, *ICA*, applied to the binary web page classification task. Each step in the sequence displays a graph of four nodes, where each node denotes a web page, and hyperlinks among them. Each node has a class label y_i; the set of possible class labels is $C = \{P, S\}$, denoting *professors* and *students*, respectively. Three nodes have *unknown* labels ($V^U = \{v_1, v_2, v_4\}$) and one node has a *known* label ($V^K = \{v_3\}$). In the initial state (step A), no label y_i has yet been estimated for the nodes in V^U, so each is set to *missing* (indicated by a question mark). Each node has three binary attributes (represented by $\vec{x}_i$). Nodes in V^U also have two relational features (one per class), represented by the vector $\vec{f}_i$. Each feature denotes the number of linked nodes (ignoring link direction) that have a particular class label.

In step B, some classifier (not shown) estimates class labels for nodes in V^U using only the (non-relational) attributes. These labels, along with the known label y_3, are used in step C to compute the relational feature value vectors. For instance, in step C, $\vec{f}_2 = (1\ 2)$ because v_2 links to nodes with one current *P* label and two current *S* labels. In step D, a classifier re-estimates the labels using both attributes and relational features, which changes the predicted label of v_2. In step E, relational feature values are re-computed using the new labels. Steps D and E then repeat until a termination criterion is satisfied (e.g., convergence, number of iterations).

This example exhibits how relational value uncertainty occurs with CC. For instance, the feature vector $\vec{f}_1$ is $(1\ 0)$ in step C but later becomes $(0\ 1)$. Thus, intermediate predictions use uncertain label estimates, motivating the need to cautiously use such estimates.

1. V^K may be empty. In addition, a separate training graph may be provided; see Section 2.3.

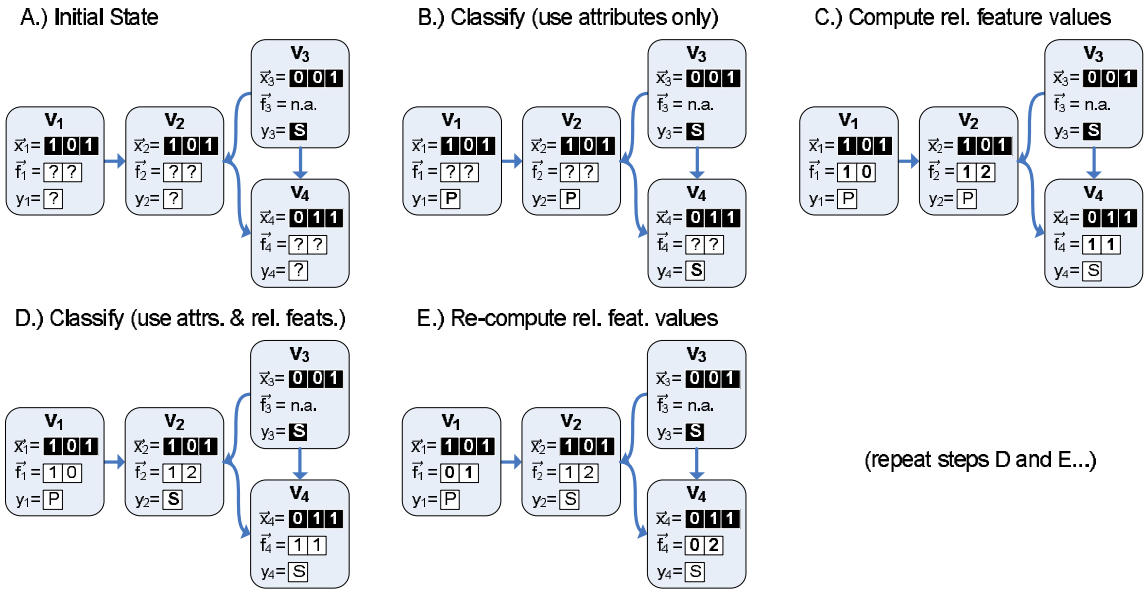


Figure 1: Example operation of *ICA*, a simple CC algorithm. Each step (A thru E) shows a graph of 4 linked nodes (i.e., web pages). "Known" values are are shown in white text on a black background; this includes all attribute values $\vec{x}_i$ and the class label y_3 for v_3. Estimated values are shown instead with a white background.

2.2 Algorithms for Collective Inference

For some collective inference tasks, exact methods such as junction trees (Huang and Darwiche, 1996) or variable elimination (Zhang and Poole, 1996) can be applied. However, these methods may be prohibitively expensive to use (e.g., summing over the remaining variable configurations is intractable for modest-sized graphs). Some research has focused on methods that further factorize the variables, and then apply an exact procedure such as belief propagation (Neville and Jensen, 2005), min-cut partition (Barzilay and Lapata, 2005), or methods for solving quadratic and linear programs (Triebel et al., 2007). In this article, we consider only approximate collective inference methods.

We consider three primary types of approximate collective inference algorithms, borrowing some terminology from Sen et al. (2008):

- **Local classifier-based methods.** For these methods, inference is an iterative process whereby a *local classifier* predicts labels for each node in V^U using both attributes and relational features (derived from the current label predictions), and then a *collective inference* algorithm recomputes the class labels, which will be used in the next iteration. Examples of this type of CC algorithm include *ICA* (used in the example above) and *Gibbs*. Local classifiers that have been used include Naive Bayes (Jensen et al., 2004), relational probability trees (Neville et al., 2003a), k-nearest neighbor (McDowell et al., 2007b), and logistic regression (Sen et al., 2008). Typically, a supervised learner induces the local classifier from the training set using both attributes and relational features.

- **Global formulation-based methods.** These methods train a classifier that seeks to optimize one global objective function, often based on a Markov random field (Dobrushin, 1968; Besag, 1974). As above, the classifier uses both attributes and relational features for inference. Examples of these algorithms include loopy belief propagation and relaxation labeling. These do not use a separate local classifier; instead, the entire algorithm is used for both training (e.g., to learn the clique potentials) and inference. See Taskar et al. (2002) and Sen et al. (2008) for more details.

- **Relational-only methods.** Recently, Macskassy and Provost (2007) demonstrated that, when some labels are known (i.e., $|V^K| > 0$), algorithms that use *only* relational information can in some cases perform very well. We consider several variants of the algorithm they described, $wvRN_{RL}$ (weighted-vote relational neighbor, with relaxation labeling). This algorithm computes a new label distribution for a node by averaging the current distributions of its neighbors. It does not require any training.

With local classifier methods, learning the classifier can often be done in a single pass over the data, does not require running collective inference, and in fact is independent of the collective inference procedure that will be used. In contrast, for global methods the local classifier and inference algorithm are effectively unified. As a result, learning for a global method requires committing to and actually executing a specific inference algorithm, and thus can be much slower than with a local classifier-based method.

All of these algorithms jointly classify interrelated nodes using some iterative process. Those that propagate from one iteration to the next a single label for each node are called *hard-labeling* methods. Methods that instead propagate a probability distribution over the possible class labels are called *soft-labeling* methods (cf., Galstyan and Cohen, 2007). All of the local classifier-based methods that we examine are hard-labeling methods.[2] Soft-labeling methods, such as variants of relaxation labeling, are also possible but require that the local classifier be able to reason directly with label distributions, which is more complex than the label aggregation for features typically done with approaches like *ICA* or *Gibbs*. Section 6.6 provides more detail on these features.

2.3 Task Definitions and Focus

Collective classification has been applied to two types of inference tasks, namely the **out-of-sample** task, where V^K is empty, and the **in-sample** task, where V^K is not empty. Both types of tasks may emerge in real-world situations (Neville and Jensen, 2005). Prior work on out-of-sample tasks (Neville and Jensen, 2000; Taskar et al., 2002; Sen and Getoor, 2006) assume that the algorithm is also provided with a training graph G_{Tr} that is disjoint from the test graph G. For instance, a model may be learned over the web-graph for one institution, and tested on the web-graph of another.

For in-sample tasks, where some labels in G are known, CC can be applied to the single graph G (Macskassy and Provost, 2007; McDowell et al., 2007a; Sen et al., 2008; Gallagher et al., 2008); *within-network* classification (Macskassy and Provost, 2006) involves training on the subset $G^K \subset G$ with known labels, and testing by running inference over the entire graph. This task simulates, for example, fraud detection in a single large telecommunication network where some entities/nodes are

2. We could also consider $wvRN_{RL}$, which is a soft-labeling method, to be a local classifier-based method, albeit a simple one that ignores attributes and does no learning. However, for explication we list relational-only methods as a separate category in the list above because our results will show they often have rather different performance trends.

known to be fraudulent. Another in-sample task (Neville and Jensen, 2007; Bilgic and Getoor, 2008; Neville and Jensen, 2008) assumes a separate training graph G_{Tr}, where a model is learned from G_{Tr} and inference is performed over the test graph G, which includes both labeled and unlabeled nodes. For both tasks, predictive accuracy is measured only for the unlabeled nodes.

In Section 6, we will address three types of tasks (i.e., out-of-sample, sparse in-sample, and dense in-sample). This is similar to the set of tasks addressed in some previous evaluations (e.g., Neville and Jensen, 2007, 2008; Bilgic and Getoor, 2008) and subsumes some others (e.g., Neville and Jensen, 2000; Taskar et al., 2002; Sen and Getoor, 2006). We will not directly address the within-network task, but the algorithmic trends observed from our in-sample evaluations should be similar.[3]

2.4 Assumptions and Limitations

In this broad investigation on the utility of caution in collective classification, we make several simplifying assumptions. First, we assume data is obtained passively rather than actively (Rattigan et al., 2007; Bilgic and Getoor, 2008). Second, we assume that nodes are homogeneous (e.g., all represent the same kind of object) rather than heterogeneous (Neville et al., 2003a; Neville and Jensen, 2007). Third, we assume that links are not missing, and need not be inferred (Bilgic and Getoor, 2008). Finally, we do not attempt to increase autocorrelation via techniques such as link addition (Gallagher et al., 2008), clustering (Neville and Jensen, 2005), or problem transformation (Tian et al., 2006; Triebel et al., 2007).

Our example in Figure 1 employs a simple relational feature (i.e., that counts the number of linked nodes with a specific class label). However, several other types of relations exist. For example, Gallagher and Eliassi-Rad (2008) describe a topology of feature types, including structural features that are independent of node labels (e.g., the number of linked neighbors of a given node). We focus on only three simple types of relational features (see Section 6.6), and leave broader investigations for future work. Likewise, for CC algorithms that learn, we assume that training is performed just once, which differs from some prior work where the learned model is updated in each iteration (Lu and Getoor, 2003b; Gurel and Kersting, 2005).

3. Related Work

Besag (1986) originally described the "Iterated Conditional Modes" (ICM) algorithm, which is a version of the *ICA* algorithm that we consider. Several researchers have reported that employing inter-instance relations in CC algorithms can significantly increase predictive accuracy (e.g., Chakrabarti et al., 1998; Neville and Jensen, 2000; Taskar et al., 2002; Lu and Getoor, 2003a). Furthermore, these algorithms have performed well on a variety of tasks, such as identifying securities fraud (Neville et al., 2005), ranking suspicious entities (Macskassy and Provost, 2005), and annotating semantic web services (Heß and Kushmerick, 2004).

In each iteration, a CC algorithm predicts a class label (or a class distribution) for each node and uses it to determine the next iteration's predictions. Although using label predictions from linked nodes (instead of using the larger number of attributes from linked nodes) encapsulates the influence of a linked node and simplifies learning (Jensen et al., 2004), it can be problematic. For example,

3. Indeed, we performed additional experiments where we reproduced the synthetic data of Sen et al. (2008), but then transformed the task from their within-network variant to a variant that uses a separate graph for training (as done in this article), and obtained results similar to those they reported.

iterating with incorrectly predicted labels can propagate and amplify errors (Neville and Jensen, 2007; Sen and Getoor, 2006; Sen et al., 2008), diminishing or even reducing accuracy compared to non-relational approaches. In this article, we examine the data characteristics (and algorithmic interactions) for which these issues are most serious and explain how cautious approaches can ameliorate them.

The performance of CC compared to non-relational learners depends greatly on the data characteristics. First, for CC to improve performance, the data must exhibit *relational autocorrelation* (Jensen et al., 2004; Neville and Jensen, 2005; Macskassy and Provost, 2007; Rattigan et al., 2007; Sen et al., 2008), which is correlation among the labels of related instances (Jensen and Neville, 2002). Complex correlations can be exploited by some CC algorithms, capturing for instance the notion "Professors primarily have out-links to Students." In contrast, the simplest kind of correlation is *homophily* (McPherson et al., 2001), in which links tend to connect nodes with the same label. To facilitate replication, Appendix A defines homophily more formally.

A second data characteristic that can influence CC performance is *attribute predictiveness*. For example, if the attributes are far less predictive than the selected relational features, then CC algorithms should perform comparatively well vs. traditional algorithms (Jensen et al., 2004). Third, *link density* plays a role (Jensen and Neville, 2002; Neville and Jensen, 2005; Sen et al., 2008); if there are few relations among the instances, then collective classification may offer little benefit. Alternatively, algorithms such as *LBP* are known to perform poorly when link density is very high (Sen and Getoor, 2006). Fourth, an important factor is the *labeled proportion* (the proportion of test nodes that have known labels). In particular, if some node labels are known ($|V^K| > 0$), these labels may help prevent CC estimation errors from cascading. In addition, if a substantial number of labels are known, simpler relational-only algorithms may be the most effective. Although additional data characteristics exist that can influence the performance of CC algorithms, such as *degree of disparity* (Jensen et al., 2003) and *assortativity* (Newman, 2003; Macskassy, 2007), we concentrate on these four in our later evaluations.

Compared to this article, prior studies provide complementary results and make some relevant comparisons, but do not examine important variations that we consider here. For instance, Jensen et al. (2004) only investigate a single collective inference algorithm, and Macskassy and Provost (2007) focus on relational-only (univariate) algorithms. Sen et al. (2008) assess several algorithms on real and synthetic data, but do not examine the impact of attribute predictiveness or labeled proportion. Likewise, Neville and Jensen (2007) evaluate synthetic and real data, but vary data characteristics (autocorrelation and labeled proportion) for only the synthetic data, do not consider *ICA*, and consider *LBP* only for the synthetic data. In addition, only one of these prior studies (Neville and Jensen, 2000) evaluates an algorithm related to ICA_C, which is a simple cautious variant of *ICA* that we show has promising performance. Moreover, these studies did not compare algorithms that vary only in their degree of cautious inference, or use cautious learning.

4. Types of Caution in CC and Why Caution is Important

Section 3 described how collective classification exploits label predictions to try to increase accuracy, but how iterating with incorrectly predicted labels can sometimes propagate and amplify errors. To address this problem, we recently proposed the use of cautious inference for CC (McDowell et al., 2007a). We defined an inference algorithm to be cautious if it sought to "explicitly identify and preferentially exploit the more certain relational information." In addition, we ex-

plained that a variant of *ICA* that we here call ICA_C is cautious because it selectively ignores class labels that were predicted with less confidence by the local classifier. Previously, Neville and Jensen (2000) introduced a simpler version[4] of ICA_C but compared it only with non-relational classifiers. We showed that ICA_C can outperform *ICA* and *Gibbs*, but did not identify the data conditions under which such gains hold.

In this article, we expand our original notion of caution in two ways. First, we broaden our idea of *cautious inference* to encompass several other existing CC inference techniques that seek the same goal (managing prediction uncertainty). Recognizing the behavioral similarities between these different algorithms helps us to better assess the strengths and weaknesses of each algorithm for a particular data set. Second, we introduce *cautious learning*, a technique that ameliorates prediction uncertainty even before inference is applied, which can substantially increase accuracy. Below we detail these two types of caution.

- A CC algorithm exhibits **cautious inference** if its inference process attends to the uncertainty of its intermediate label predictions. Usually, this uncertainty is approximated via the posterior probabilities associated with each predicted label. For instance, a CC algorithm may exercise cautious inference by favoring predicted information that has less uncertainty (higher confidence). This is the approach taken by ICA_C, which uses only the most certain labels at the beginning of its operation, then gradually incorporates less certain predictions in later iterations. Alternatively, instead of always selecting the most likely class label for each node (like *ICA* and ICA_C), *Gibbs* re-samples the label of each node based on its estimated distribution. This re-sampling leads to more stochastic variability (and less influence) for nodes with less certain predictions. Finally, soft-labeling algorithms like *LBP*, relaxation labeling, and $wvRN_{RL}$ directly reason with the estimated label distributions. For instance, $wvRN_{RL}$ averages the estimated distributions of a node's linked neighbors, which gives more influence to more certain predictions.

- A CC algorithm exhibits **cautious learning** if its training process is influenced by recognizing the disparity between the training set (where labels are known and certain) and the test scenario (where labels may be estimated and hence incorrect). In particular, a relational feature may appear to be highly predictive of the class when examining the training set (e.g., to learn conditional probabilities or feature weights), yet its use may actually decrease accuracy if its value is often incorrect during testing. In response, one approach is to ensure that appropriate training parameters are cross-validated using the actual testing conditions (e.g., with estimated test labels). We use PLUL to achieve this goal.

The next section describes how these general ideas can be applied. Later, our experimental results demonstrate when they lead to significant performance improvements.

5. Applying Cautious Inference and Learning to Collective Classification

The previous section described two types of caution for CC. Each attempts to alleviate potential estimation errors in labels during collective inference. Cautious inference and cautious learning can often be combined, and at least one is used or is applicable to every CC algorithm known to

4. Their algorithm is like ICA_C, except that it does not consider how to favor "known" labels from V^K.

us. In this section, we provide examples of how both types can be applied by describing specific CC algorithms that exploit cautious inference (Sections 5.1-5.4), and by describing how PLUL can complement these algorithms with cautious learning (Section 5.5). Section 5.6 then discusses the computational complexity of these algorithms.

We describe and evaluate four families of CC inference algorithms: *ICA*, *Gibbs*, *LBP*, and *wvRN*.[5] Among local classifier-based algorithms, we chose *ICA* and *Gibbs* because both have been frequently studied and often perform well. As a representative global formulation-based algorithm, we chose *LBP* instead of relaxation labeling because previous studies (Sen and Getoor, 2007; Sen et al., 2008) found similar performance, with in some cases a slight edge for *LBP*. Finally, we select *wvRN* because it is a good relational-only baseline for CC evaluations (Macskassy and Provost, 2007).

Table 1 summarizes the four CC families that we consider. Within each family, each variant use more cautious inference than the variant listed below it. Cautious variants of standard algorithms are given a "C" subscript (e.g., ICA_C), while non-cautious variants of standard algorithms are given a "NC" subscript (e.g., $Gibbs_{NC}$). For the latter case, our intent is not to demonstrate large performance "gains" for a standard algorithm vs. a new non-cautious variant, but to isolate the impact of a particular cautious algorithmic behavior on performance. While the result may not be a theoretically coherent algorithm (e.g., $Gibbs_{NC}$, unlike *Gibbs*, is not a MCMC algorithm), in every case the resultant algorithm *does* perform well under data set situations where caution is not critical (see Section 7). Thus, comparing the performance of the cautious vs. non-cautious variants allows us to investigate the data characteristics for which cautious behavior is more important.[6]

5.1 ICA Family of Algorithms

Figure 2 displays pseudocode for *ICA*, ICA_C, and ICA_{Kn}, depending on the setting of the parameter $AlgType$. We describe each in turn.

5.1.1 *ICA*

In Figure 2, step 1 is a "bootstrap" step that predicts the class label y_i of each node in V^U using only attributes ($conf_i$ records the confidence of this prediction, but *ICA* ignores this information). The algorithm then iterates (step 2). During each iteration, *ICA* selects all available labels (step 3), computes the relational features' values based on these labels (step 4), and then re-predicts the class label of each node using both attributes and relational features (step 5). After iterating, hopefully to convergence, step 6 returns the final set of estimated class labels.

Types of Caution Used: Steps 3-4 of *ICA* use all available labels for feature computation (including estimated, possibly incorrect labels) and step 5 picks the single most likely label for each node based on the new predictions. In these steps, uncertainty in the predictions is ignored. Thus, *ICA* does not

5. Technically, *wvRN* by itself is a local classifier, not an inference algorithm, but for brevity we refer to the family of algorithms based on this classifier (such as $wvRN_{RL}$) as *wvRN*.
6. Section 7 shows that the non-cautious variants *ICA*, $Gibbs_{NC}$, and LBP_{NC} perform similarly to each other. Thus, our empirical results would change little if we compared all of the cautious algorithms against the more standard *ICA*. However, the results for *Gibbs* and *LBP* would then concern performance differences between distinct algorithms, due to conjectured but unconfirmed differences in algorithmic properties. By instead comparing *Gibbs* vs. $Gibbs_{NC}$ and *LBP* vs. LBP_{NC}, we more precisely demonstrate that the cautious algorithms benefit from specifically identified cautious behaviors.

Name	Cautious Inf.?	Key Features	Type	Evaluated by?
Local classifier-based methods that iteratively classify nodes, yielding a final graph state				
ICA_C	Favors more conf. labels	Relational features depend only on "more confident" estimated labels; later iterations loosen confidence threshold.	Hard	Neville and Jensen (2000); McDowell et al. (2007a)
ICA_{Kn}	Favors known labels	First iteration: rel. features depend only on known labels. Later iterations: use all labels.	Hard	McDowell et al. (2007a)
ICA	Not cautious	Always use all labels, known and estimated.	Hard	Lu and Getoor (2003a); Sen and Getoor (2006); McDowell et al. (2007a,b)
Local classifier-based algorithms that compute conditional probabilities for each node				
Gibbs	Samples from estimated distribution	At each step, classifies using *all* neighbor labels, then samples new labels from the resultant distributions. Records new labels to produce final marginal statistics.	Hard	Jensen et al. (2004); Neville and Jensen (2007); Sen et al. (2008)
$Gibbs_{NC}$	Not cautious	Like *Gibbs*, but always pick most likely label instead of sampling.	Hard	None, but very similar to ICA.
Global formulation algorithms based on loopy belief propagation (LBP)				
LBP	Reasons with estimated distribution	Passes continuous-valued messages between linked neighbors until convergence.	Soft	Taskar et al. (2002); Neville and Jensen (2007); Sen et al. (2008)
LBP_{NC}	Not cautious	Like *LBP*, but each node always chooses single most likely label to use for next round of messages.	Hard	—
Relational-only algorithms				
$wvRN_{RL}$	Reasons with estimated distribution	Computes new distribution by averaging neighbors' label distributions; combines old and new distributions via relaxation labeling.	Soft	Macskassy and Provost (2007); Gallagher et al. (2008)
$wvRN_{ICA+C}$	Favors nodes closer to known labels	Initializes nodes in V^U to *missing*. Computes most likely label by averaging neighbors' labels, ignoring *missing* labels.	Hard	Macskassy and Provost (2007); similar to Galstyan and Cohen (2007)
$wvRN_{ICA+NC}$	Not cautious	Like $wvRN_{ICA+C}$, but no *missing* labels are used. Instead, initialize nodes in V^U by sampling from the prior label distribution.	Hard	—

Table 1: The ten collective inference algorithms considered in this article, divided into four families. Hard/soft refers to hard-labeling and soft-labeling (see Section 2.2).

perform cautious inference. However, it may exploit cautious learning to learn the classifier models that are used for inference (M_A and M_{AR}).

5.1.2 ICA_C

In steps 3-4 of Figure 2, *ICA* assumes that the estimated node labels are all equally likely to be correct. When *AlgType* instead selects ICA_C, the inference becomes more cautious by only considering more certain estimated labels. Specifically, step 3 "commits" into Y' only the best m of

ICA_classify $(V, E, X, Y^K, M_{AR}, M_A, n, AlgType)$=
// V=nodes, E=edges, X=attribute vectors, Y^K=labels of known nodes ($Y^K = \{y_i | v_i \in V^K\}$)
// M_{AR}=local classifier (uses attributes and relations), M_A=classifier that uses only attributes
// n=# of iterations, $AlgType$=ICA_C, ICA_{Kn}, or ICA

1 **for each** node $v_i \in V^U$ **do** // Bootstrap: estimate label y_i for each node
  $(y_i, conf_i) \leftarrow M_A(\vec{x}_i)$ // using attributes only
2 **for** $h = 0$ **to** n **do**
3 // Select node labels to use for computing relational feature values, store in Y'
  if $(AlgType = ICA_C)$ // For ICA_C: use known and m most confident
    $m \leftarrow |V^U| \cdot (h/n)$ // estimated labels, gradually increase m
    $Y' \leftarrow Y^K \cup \{y_i | v_i \in V^U \wedge rank(conf_i) \leq m\}$
  else if $(h = 0)$ and $(AlgType = ICA_{Kn})$
    $Y' \leftarrow Y^K$ // For ICA_{Kn}(first iteration): use *only* known labels
  else // For ICA_{Kn} (after first iteration) and ICA: use *all*
    $Y' \leftarrow Y^K \cup \{y_i | v_i \in V^U\}$ // labels (known and estimated)
4 **for each** node $v_i \in V^U$ **do**
    $\vec{f}_i \leftarrow calcRelatFeats(V, E, Y')$ // Compute feature values, using labels selected above
5 **for each** node $v_i \in V^U$ **do** // Re-predict most likely label, using attributes
    $(y_i, conf_i) \leftarrow M_{AR}(\vec{x}_i, \vec{f}_i)$ // and relational features
6 **return** $\{y_i | v_i \in V^U\}$ // Return predicted class label for each node

Figure 2: Algorithm for *ICA* family of algorithms. We use $n = 10$ iterations.

the current estimated labels; other labels are considered *missing* and thus ignored in the next step. Step 4 computes the relational features using only the committed labels, and step 5 classifies using this information. Step 3 gradually increases the fraction of estimated labels that are committed per iteration (e.g., if n=10, from 0%, 10%, 20%,..., up to 100%). Node label assignments committed in an iteration h are not necessarily committed again in iteration $h+1$ (and may in fact change).

ICA_C requires some kind of confidence measure ($conf_i$ in Figure 2) to determine the "best" m of the current label assignments (those with the highest confidence "rank"). We adopt the approach of Neville and Jensen (2000) and use the posterior probability of the most likely class for each node i as $conf_i$. In exploratory experiments, we found that alternative measures (e.g., probability difference of the top two classes) produced similar results.

Types of Caution Used: ICA_C favors more confident information by ignoring nodes whose labels are estimated with lower confidence. Step 3 executes this preference, which affects the algorithm in several ways. First, omitting the estimated labels for some nodes causes the relational feature value computation in step 4 to ignore those less certain labels. Since this computation favors more reliable label assignments, subsequent assignments should also be more reliable. Second, if any node links only to nodes with *missing* labels, then the computed value of the relational features for that node will also be *missing*; Section 6.5 describes how the classifier in Step 5 handles this case. Third, recall that a realistic CC scenario's test set may have links to nodes with known labels; these nodes, represented by V^K, provide the "most certain" labels and thus may aid classification. ICA_C exploits *only* these labels for iteration $h = 0$. In this case, step 3 ignores all estimated labels (every estimate for V^U), but step 4 can still compute some relational feature values based on known labels

Gibbs_classify $(V, E, X, Y^K, M_{AR}, M_A, n, n_B, C, AlgType)$=
// V=nodes, E=edges, X=attribute vectors, Y^K=labels of known nodes ($Y^K = \{y_i | v_i \in V^K\}$)
// M_{AR}=local classifier (uses attributes and relations), M_A=classifier that uses only attributes
// n=# of iterations, n_B= "burn-in" iters., C=set of class labels, $AlgType$=*Gibbs* or $Gibbs_{NC}$

1 **for each** node $v_i \in V^U$ **do** // Bootstrap: estimate label probs. $\vec{b}_i$
  $\vec{b}_i \leftarrow M_A(\vec{x}_i)$ // for each node, using attributes only
2 **for each** node $v_i \in V^U$ **do** // Initialize statistics
  for each $c \in C$
    $stats[i][c] \leftarrow 0$
3 **for** $h = 1$ **to** n **do** // Repeat for n iterations
  for each node $v_i \in V^U$ **do**
4 **switch** ($AlgType$):
    case ($Gibbs$): $y_i \leftarrow sampleDist(\vec{b}_i)$ // Sample next label from distribution
    case ($Gibbs_{NC}$): $y_i \leftarrow argmax_{c \in C}\ b_i(c)$ // Or, pick most likely label from dist.
5 **if** $(h > n_B)$ $stats[i, y_i] \leftarrow stats[i, y_i] + 1$ // Record stats. on chosen label
6 $Y' \leftarrow Y^K \cup \{y_i | v_i \in V^U\}$ // Compute feature values, using known
  for each node $v_i \in V^U$ **do** // labels and labels chosen in step 4
    $\vec{f}_i \leftarrow computeRelatFeatures(V, E, Y')$
7 **for each** node $v_i \in V^U$ **do** // Re-estimate label probs., using
    $\vec{b}_i \leftarrow M_{AR}(\vec{x}_i, \vec{f}_i)$ // attributes and relational features
8 **return** *stats* // Return marginal stats. for each node

Figure 3: Algorithm for Gibbs sampling. Thousands of iterations are typically needed.

from V^K. Thus, the known labels influence the first classification in step 5, before any estimated labels are used, and in subsequent iterations. Finally, ICA_C can also benefit from PLUL.

5.1.3 ICA_{Kn}

The above discussion highlighted two different effects from ICA_C: favoring more confident estimated labels vs. favoring known labels from V^K. An interesting variant is to favor the known labels in the first iteration (just like ICA_C), but then use all labels for subsequent iterations (just like *ICA*). We call this algorithm ICA_{Kn} ("ICA+Known").

Types of Caution Used: ICA_{Kn} favors only known nodes. It is thus more cautious than *ICA*, but less cautious than ICA_C. It can also benefit from cautious learning via PLUL.

5.2 Gibbs Family of Algorithms

Figure 3 displays pseudocode for Gibbs sampling (*Gibbs*) and the non-cautious variant $Gibbs_{NC}$. We describe each in turn.

5.2.1 *Gibbs*

In Figure 3, step 1 (bootstrapping) is identical to step 1 of the *ICA* algorithms, except that for each node v_i the classifier must output a distribution $\vec{x}_i$ containing the likelihood of each class, not just

the most likely class. Step 2 initializes the statistics that will be used to compute the marginal class probabilities for each node. In step 4, within the loop, the algorithm probabilistically samples the current class label distribution of each node and assigns a single label y_i based on this distribution. This label is also recorded in the statistics during Step 5 (after the first n_B iterations are ignored for "burn-in"). Step 6 then selects all labels (known labels and those just sampled) and uses them to compute the relational features' values. Step 7 re-estimates the posterior class label probabilities given these relational features. The process then repeats. When the process terminates, the statistics recorded in step 5 approximate the marginal distribution of class labels, and are returned by step 8.
Types of Caution Used: Like ICA_C, *Gibbs* is cautious in its use of estimated labels, but in a different way. In particular, ICA_C exercises caution in step 3 by ignoring (at least for some iterations) labels that have lower confidence. In contrast, *Gibbs* exercises caution by sampling, in step 4, values from each node's predicted label distribution—causing nodes with lower prediction confidence to reflect that uncertainty via higher fluctuation in their assigned labels, yielding less predictive influence on their neighbors. Gibbs can also benefit from cautious learning via PLUL.

We expect *Gibbs* to perform better than ICA_C, since it makes use of more information, but this requires careful confirmation. In addition, *Gibbs* is considerably more time intensive than ICA_C or *ICA* (see Section 5.6).

5.2.2 $Gibbs_{NC}$

$Gibbs_{NC}$ is identical to *Gibbs* except that instead of sampling in step 4, it always selects the most likely label. This change makes $Gibbs_{NC}$ deterministic (unlike *Gibbs*), and makes $Gibbs_{NC}$ behave almost identically to *ICA*. In particular, observe that after any number of iterations h ($1 \leq h \leq n$), *ICA* and $Gibbs_{NC}$ will have precisely the same set of current label assignments for every node. However, *ICA*'s result is the final set of label assignments, whereas $Gibbs_{NC}$'s result is the marginal statistics computed from these time-varying assignments. For a given data set, if *ICA* converges to an an unchanging set of label assignments, then for sufficiently large n $Gibbs_{NC}$'s final result (in terms of accuracy) will be identical to *ICA*'s. If, however, some nodes' labels continue to oscillate with *ICA*, then *ICA* and $Gibbs_{NC}$ will have different results for some of those nodes.
Types of Caution Used: Just like *ICA*, $Gibbs_{NC}$ uses all available labels for relational feature computation, and always picks the single most likely label based on the new predictions. Thus, $Gibbs_{NC}$ does not perform cautious inference, though it can benefit from cautious learning to learn the classifiers M_A and M_{AR}.

5.3 LBP Family of Algorithms

This section describes loopy belief propagation (*LBP*) and the non-cautious variant LBP_{NC}.

5.3.1 *LBP*

LBP has been a frequently studied technique for performing approximate inference, and has been used both in early work on CC (Taskar et al., 2002) and in more recent evaluations (Sen and Getoor, 2006; Neville and Jensen, 2007; Sen et al., 2008). Most works that study *LBP* for CC treat the entire graph, including attributes, as a pairwise Markov random field (e.g., Sen and Getoor, 2006; Sen et al., 2008) and then justify *LBP* as an example of a variational method (cf., Yedidia et al., 2000). The basic inference algorithm is derived from belief propagation (Pearl, 1988), but applied to graphs with cycles (McEliece et al., 1998; Murphy et al., 1999).

LBP performs inference via passing messages from node to node. In particular, $m_{i\to j}(c)$ represents node v_i's assessment of how likely it is that node v_j has a true label of class c. In addition, $\phi_i(c)$ represents the "non-relational evidence" (e.g., based only on attributes) for v_i having class c, and $\psi_{ij}(c',c)$ represents the "compatibility function" which describes how likely two nodes of class c and c' are linked together (in terms of Markov networks, this represents the potential functions defined by the pairwise cliques of linked class nodes). Given these two sets of functions, Yedidia et al. (2000) show the belief that node i has class c can be calculated as follows:

$$b_i(c) = \alpha\phi_i(c)\prod_{k\in N_i} m_{k\to i}(c) \tag{1}$$

where α is a normalizing factor to ensure that $\sum_{c\in C} b_i(c) = 1$ and N_i is the neighborhood function defined as:

$$N_i = \{v_j | \exists (v_i, v_j) \in E\} .$$

The messages themselves are computed recursively as:

$$m'_{i\to j}(c) = \alpha \sum_{c'\in C}\left(\phi_i(c')\psi_{ij}(c',c)\prod_{k\in N_i\setminus j} m_{k\to i}(c')\right) . \tag{2}$$

Observe that the message from i to j incorporates the beliefs of all the neighbors of i (N_i) *except* j itself. $m'_{i\to j}(c)$ is the "new" value of $m_{i\to j}(c)$ to be used in the next iteration.

For CC, we need a model that generalizes from the training nodes to the test nodes. The above equations do not provide this, since they have node-specific potential functions (i.e., ψ_{ij} is specific to nodes i and j). Fortunately, we can represent each potential function as a log-linear combination of generalizable features, as commonly done for such Markov networks (e.g., Della Pietra et al., 1997; McCallum et al., 2000a). More specifically for CC, Taskar et al. (2002) used a log-linear combination of functions that indicate the presence or absence of particular attributes or other features. Several papers (e.g., Sen and Getoor, 2006; Sen et al., 2008) have described a general model on how to accomplish this, but do not completely explain how to perform the computation. For a slight loss in generality (e.g., assuming that our nodes are represented by a simple attribute vector), we now describe how to perform *LBP* for CC on an undirected graph. In particular, let N_A be the number of attributes, $\mathcal{D}_h$ be the domain of attribute h, and $w_{c,h,k}$ be a learned weight indicating how strongly a value of k for attribute h indicates that a given node has class c. In addition, let $f_i(h,k)$=1 iff the h^{th} attribute of node i is k (i.e., $x_{ih} = k$). Then

$$\phi_i(c) = exp\left(\sum_{h\in\{1..N_A\}}\sum_{k\in\mathcal{D}_h} exp(w_{c,h,k})f_i(h,k))\right)$$

which is a special case of logistic regression. We likewise define similar learned weights of the form $w_{c,c'}$ that indicate how likely a node with label c is linked to a node with label c', yielding the compatibility function

$$\psi_{ij}(c,c') = exp(w_{c,c'}) .$$

LBP_classify $(V,E,X,Y^K,w,C,N,AlgType)$=
// V=nodes, E=edges, X=attribute vectors, Y^K=labels of known nodes ($Y^K = \{y_i | v_i \in V^K\}$)
// w=learned params., C=set of class labels, N=neighborhood funct., $AlgType$=*LBP* or LBP_{NC}

1 **for each** $(v_i, v_j) \in E$ such that $v_j \in V^U$ **do** // Initialize all messages
  for each $c \in C$ **do**
    if $(v_i \in V^K)$ // If class is known (y_i), set message to its
      $m_{i\to j}(c) \leftarrow \alpha \cdot exp(w_{y_i,c})$ // final, class-specific value
    else // Otherwise, message starts with same value
      $m_{i\to j}(c) \leftarrow \alpha$ // for every class, but will vary later

2 **while** (messages are still changing)

3 **for each** $(v_i, v_j) \in E$ such that $v_j \in V^U$ **do** // Perform message passing
    for each $c \in C$ **do**
      $m'_{i\to j}(c) \leftarrow \alpha \sum_{c' \in C} \left(\phi_i(c') exp(w_{c',c}) \prod_{k \in N_i \setminus j} m_{k\to i}(c')\right)$

4 **for each** $(v_i, v_j) \in E$ such that $v_j \in V^U$ **do**
    if $(AlgType = LBP)$ // For *LBP*, copy new messages for use in
      $m_{i\to j}(c) \leftarrow m'_{i\to j}(c)$ // next iteration
    else
      $c' \leftarrow argmax_{c \in C}(m'_{i\to j}(c))$ // For LBP_{NC}, select most likely label for node
      for each $c \in C$ **do** // Treat selected label the same as a "known"
        $m_{i\to j}(c) \leftarrow exp(w_{c',c})$ // label for use in the next iteration

5 **for each** node $v_i \in V^U$ **do** // Compute final beliefs
  for each $c \in C$ **do**
    $b_i(c) \leftarrow \alpha \phi_i(c) \prod_{k \in N_i} m_{k\to i}(c)$

6 **return** $\{\vec{b}_i\}$ // Return final beliefs

Figure 4: Algorithm for loopy belief propagation (LBP). α is a normalization factor.

As desired, the compatibility function is now independent of specific node identifiers, that is, it depends only upon the class labels c and c', not i and j. We use conjugate gradient descent to learn the weights (cf., Taskar et al., 2002; Neville and Jensen, 2007; Sen et al., 2008).

Finally, we must consider how to handle messages from nodes with a "known" class label. Suppose node v_i has known class y_i. This is equivalent to having a node where the non-relational evidence $\phi_i(c) = 1$ if c is y_i and zero otherwise. Since y_i is known, node v_i is not influenced by its neighbors. In that case, using Equation 2 (with an empty neighborhood for the product) yields:

$$m_{i\to j}(c) = \alpha \sum_{c' \in C} \phi_i(c') \psi_{ij}(c', c) = \alpha \cdot \psi_{ij}(y_i, c) = \alpha \cdot exp(w_{y_i,c}) \; . \tag{3}$$

Given these formulas, we can now present the complete algorithm in Figure 4. In Step 1, the messages are initialized, using Equation 3 if v_i is a known node; otherwise, each value is set to α (creating a uniform distribution). Steps 2-4 performs message passing until convergence, based on Equation 2. Finally, step 5 computes the final beliefs using Equation 1 and step 6 returns the results.
Types of Caution Used: Like *Gibbs*, *LBP* exercises caution by reasoning based on the estimated label uncertainty, but in a different manner. Instead of sampling from the estimated distribution, *LBP* in step 3 directly updates its beliefs using all of its current beliefs, so that the new beliefs reflect the underlying uncertainty of the old beliefs. In particular, this uncertainty is expressed

WVRN_RL_classify $(V, Y^K, n, C, \vec{b}_{prior}, N, \Gamma)$=
// V=nodes, Y^K=labels of known nodes ($Y^K = \{y_i | v_i \in V^K\}$), n=# of iterations
// C=set of class labels, $\vec{b}_{prior}$=class priors, N=neighborhood funct., Γ=decay factor

1 **for each** node $v_i \in V^K$ **do** // Create belief vector for each known label
  $\vec{b}_i \leftarrow makeBeliefsFromKnownClass(|C|, y_i)$ // (all zeros except at index for class y_i)
  for each node $v_i \in V^U$ **do** // Create initial beliefs for unknown labels
  $\vec{b}_i \leftarrow \vec{b}_{prior}$ // (using class priors as initial setting)
2 **for** $h = 0$ **to** n **do** // Iteratively re-compute beliefs
3 **for each** node $v_i \in V^U$ **do** // Compute new distribution for each node
  $\vec{b'}_i \leftarrow \frac{1}{|N_i|} \sum_{v_j \in N_i} \vec{b}_j$ // by averaging neighbors' distributions
4 **for each** node $v_i \in V^U$ **do** // Perform simulated annealing
  $\vec{b}_i \leftarrow \Gamma^h \vec{b'}_i + (1 - \Gamma^h)\vec{b}_i$
5 **return** $\{\vec{b}_i | v_i \in V^U\}$ // Return belief distribution for each node

Figure 5: Algorithm for $wvRN_{RL}$. Based on Macskassy and Provost (2007), we use $n = 100$ iterations with a decay factor of $\Gamma = 0.99$.

by the continuous-valued numbers that represent each message $m_{i \to j}$. *LBP* can also benefit from cautious learning with PLUL; in this case, PLUL influences the $w_{c,h,k}$ and $w_{c,c'}$ weights that are learned (see Section 6.4).

5.3.2 LBP_{NC}

LBP_{NC} is identical to *LBP* except that after the new messages are computed in step 3, in step 4 LBP_{NC} picks the single most likely label c' to represent the message from v_i to v_j. LBP_{NC} then treats c' as equivalent to a "known" label y_i for v_i and re-computes the appropriate message $m_{i \to j}(c)$ using Equation 3.

Types of Caution Used: Like *ICA* and $Gibbs_{NC}$, LBP_{NC} is non-cautious because it uses all available labels for relational feature computation and always picks the single most likely label based on the new predictions. In essence, the "pick most likely" step transforms the soft-labeling *LBP* algorithm into the hard-labeling LBP_{NC} algorithm, removing cautious inference just as the "pick most likely" step did for $Gibbs_{NC}$. However, LBP_{NC}, like *LBP*, can still benefit from cautious learning with PLUL.

5.4 wvRN Family of Algorithms

Figure 5 displays pseudocode for $wvRN_{RL}$, a soft-labeling algorithm. For simplicity, we present the related, hard-labeling variants $wvRN_{ICA+C}$ and $wvRN_{ICA+NC}$ separately in Figure 6. Each of these is a relational-only algorithm; Section 7.9 will discuss variants that incorporate attribute information.

5.4.1 $wvRN_{RL}$

$wvRN_{RL}$(Weighted-Vote Relational Neighbor, with relaxation labeling) is a relational-only CC algorithm that Macskassy and Provost (2007) argued should be considered as a baseline for all CC

WVRN_ICA_classify $(V, Y^K, n, C, \vec{b}_{prior}, N, AlgType)$=
// V=nodes, Y^K=labels of known nodes ($Y^K = \{y_i | v_i \in V^K\}$), n=# of iters., C=class labels
// $\vec{b}_{prior}$=class priors, N=neighborhood function, $AlgType$=$wvRN_{ICA+C}$ or $wvRN_{ICA+NC}$

1 **for each** node $v_i \in V^U$ **do**
  switch ($AlgType$): // Set initial value for unknown labels...
    case ($wvRN_{ICA+C}$): $y_i \leftarrow$ '?' // ...start labels as *missing*
    case ($wvRN_{ICA+NC}$): $y_i \leftarrow sampleDist(\vec{b}_{prior})$ // ...or sample label from class priors
2 **for** $h = 0$ **to** n **do** // Iteratively re-label the nodes
3 **for each** node $v_i \in V^U$ **do**
    $N'_i \leftarrow \{v_j \in N_i | y_j \neq$ '?' $\}$ // Find all non-*missing* neighbors
    if ($|N'_i| > 0$) // New label is the most common label
      $y'_i \leftarrow argmax_{c \in C} | \{v_j \in N'_i | y_j = c\} |$ // amongst those neighbors
    else $y'_i = y_i$ // If no such neighbors, keep same label
4 **for each** node $v_i \in V^U$ **do** // After all new labels are computed,
    $y_i \leftarrow y'_i$ // update to store the new labels
5 **return** $\{y_i | v_i \in V^U\}$ // Return est. class label for each node

Figure 6: Algorithm for $wvRN_{ICA+C}$ and $wvRN_{ICA+NC}$. This is a "hard labeling" version of $wvRN_{RL}$; each of the 5 steps corresponds to the same numbered step in Figure 5. We use $n = 100$ iterations.

evaluations. At each iteration, each node i updates its estimated class distribution by averaging the current distributions of each of its linked neighbors. $wvRN_{RL}$ ignores all attributes (non-relational features). Thus, $wvRN_{RL}$ is useful only if the test set links to some nodes with known labels to "seed" the inference process. Macskassy and Provost showed that this simple algorithm can work well if the nodes exhibit strong homophily and enough labels are known.

Step 1 of $wvRN_{RL}$ (Figure 5) initializes a belief vector for every node, using the known labels for nodes in V^K, and a class prior distribution for nodes in V^U. For each node, step 3 averages the current distributions of its neighbors, while step 4 performs simulated annealing to ensure convergence. Step 5 returns the final beliefs. For simplicity, we omit edge weights from the algorithm's description, since our experiments do not use them.

Types of Caution Used: Since $wvRN_{RL}$ computes directly with the estimated label distributions, it exercises cautious inference in the same manner as *LBP*. However, unlike the other CC algorithms, it does not learn from a training set, and thus cautious learning with PLUL does not apply.

5.4.2 $wvRN_{ICA+C}$ AND $wvRN_{ICA+NC}$

Figure 6 presents a hard-labeling alternative to $wvRN_{RL}$. Each of the five steps mirror the corresponding step in the description of $wvRN_{RL}$. In particular, for node v_i, step 3 computes the most common label among the neighbors of v_i (the hard-labeling equivalent of averaging the distributions), and step 4 commits the new labels without annealing.

However, with a hard-labeling algorithm, the initial labels for each node become very important. The simplest approach would be to initialize every node to have the most common label from the prior distribution. However, that approach could easily produce interlinked regions of labels that

that were incorrect but highly self-consistent; leading to errors even when many known labels were provided. Instead, Macskassy and Provost (2007) suggest initializing each node $v_i \in V^U$ to *missing* (indicated in Figure 6 by a question mark), a value that is ignored during calculations. They call the resulting algorithm wvRN-ICA; here we refer to it as $wvRN_{ICA+C}$. A *missing* label remains for node v_i after iteration h if during that iteration every neighbor of v_i was also *missing*.

Alternatively, a simpler algorithm is to always compute with all neighbor labels (do not initialize any to *missing*), but initialize each label in V^U by sampling from the prior distribution. We call this algorithm $wvRN_{ICA+NC}$. This process is the hard-labeling analogue of $wvRN_{RL}$'s approach: instead of initializing *each* node with the prior distribution, with $wvRN_{ICA+NC}$ sampling initializes the *entire set* so that it represents, in aggregate, the prior distribution.

Types of Caution Used: $wvRN_{ICA+NC}$ always uses the estimated label of every node, without regard for how certain that estimate is. Thus, it does not exhibit cautious inference. However, $wvRN_{ICA+C}$ does exhibit cautious inference, although this effect was not discussed by prior work with this algorithm. In particular, during the first iteration $wvRN_{ICA+C}$ uses only the certain labels from Y^K, since all nodes in Y^U are marked *missing*. These known labels are used to estimate labels for every node in V^U that is directly adjacent to some node in V^K. In subsequent iterations, $wvRN_{ICA+C}$ uses both labels from Y^K and labels from Y^U that have been estimated so far. However, the labels estimated so far are likely to be more reliable than later estimations, since the former labels are from nodes that were closer to at least one known label. Thus, in a manner similar to ICA_C's gradual commitment of labels based on confidence, $wvRN_{ICA+C}$ gradually incorporates more and more estimated labels into its computation, where more confident labels (those closer to known nodes) are incorporated sooner. This effect causes $wvRN_{ICA+C}$ to exploit estimated labels more cautiously.

5.5 Parameter Learning for Uncertain Labels (PLUL)

CC algorithms typically train a local classifier on a fully-labeled training set, then use that local classifier with some collective inference algorithm to classify the test set. Unfortunately, this results in asymmetric training and test phases: since all labels are known in the training phase, the learning process sees no uncertainty in relational feature values, unlike the reality of testing. Moreover, the classifier's training is unaffected by the type of collective inference algorithm used, and how (if at all) that collective algorithm attempts to compensate for the uncertainty of estimated labels during testing. Consequently, the learned classifier may tend to produce poor estimates of important parameters related to the relational features (e.g., feature weights, conditional probabilities). Even for CC algorithms that do not use a local classifier, but instead take a global approach that learns over the entire training graph (as with *LBP* and relaxation labeling), the same fundamental problem occurs: if autocorrelation is present, then parameters learned over the fully labeled training set tend to overstate the usefulness of relational features for testing, where estimated labels must be used.

To address these problems, we developed PLUL (Parameter Learning for Uncertain Labels). PLUL is based on standard cross-validation techniques for performing automated parameter tuning (e.g., Kohavi and John, 1997). The key novelty is not in the cross-validation mechanism, but in the selection of *which* parameters should be tuned and *why*. To use PLUL, we must first select or create an appropriate parameter that controls the amount of impact that relational features have on the resultant classifications. In principle, PLUL could search a multi-dimensional parameter space, but for tractability we select a single parameter that affects all relational features. For instance, when

PLUL_learn ($CCtype, P, lp, V_{Tr}, E_{Tr}, X_{Tr}, Y_{Tr}, V_H, E_H, X_H, Y_H$)=
// $CCtype$=CC alg. to use, P=set of parameter values to consider, lp=labeled proportion to use
// $V_{Tr}, E_{Tr}, X_{Tr}, Y_{Tr}$ = vertices, edges, attributes, and labels from training graph
// V_H, E_H, X_H, Y_H = vertices, edges, attributes, and labels from holdout graph

1 $Y'_H = keepSomeLabels(lp, Y_H)$ // Randomly select lp% of labels to keep; discard others

2 $bestParam \leftarrow \emptyset$ // Initialize variables to track best parameter so far
$bestAcc \leftarrow -1$

3 **for each** $p \in P$ **do** // Iterate over every parameter value

4 // Learn complete CC classifier from fully-labeled training data, influenced by p
$cc = learn\_CC\_classifier(CCtype, V_{Tr}, E_{Tr}, X_{Tr}, Y_{Tr}, p)$

5 // Run CC on holdout graph (with some known labels Y'_H) and evaluate accuracy
$acc \leftarrow execute\_CC\_inference(cc, V_H, E_H, X_H, Y'_H)$

6 // Remember this parameter if it's the best so far
if $(acc > bestAcc)$
$bestParam \leftarrow p$
$bestAcc \leftarrow acc$

7 **return** $bestParam$ // Return best parameter found over the holdout graph

Figure 7: Algorithm for Parameter Learning for Uncertain Labels (PLUL). The holdout graph is derived from the original training data and is disjoint from the graph that is used later for testing.

using a k-nearest-neighbor rule as the local classifier, we employ PLUL to adjust the weight w_R of relational features in the node similarity function. PLUL performs automated tuning by repeatedly evaluating different values of the selected parameter, as used by the local classifier, together with the collective inference algorithm (or the entire learned model for *LBP*). For each parameter value, accuracy is evaluated on a holdout set (a subset of the training set). PLUL then selects the parameter value that yields the best accuracy to use for testing.

Figure 7 summarizes these key steps of PLUL and some additional details. First, note that proper use of PLUL requires a holdout set that reflects the test set conditions. Thus, step 1 of the algorithm removes some or all of the labels from the holdout set, leaving only the same percentage of labels (lp%) that are expected in the test set. Second, running CC inference with a new parameter value may require re-learning the local classifier (for *ICA* or *Gibbs*) or the entire learned model (for *LBP*). This is shown in step 4 of Figure 7. Alternatively, for Naive Bayes or k-nearest-neighbor local classifiers, the existing classifier can simply be updated to reflect the new parameter value.

We expect PLUL's utility to vary based upon the fraction of known labels (lp) that are available to the test set. If there are few such labels, there is more discrepancy between the training and test environments, and hence more need to apply PLUL. However, if there are many such labels, then PLUL may not be useful.

Because almost all CC algorithms learn parameters based in some way on relational features, PLUL is widely applicable. In particular, Table 2 shows how we select an appropriate relational parameter to apply PLUL for different CC algorithms. The top of the table describes how to apply PLUL to a local classifier that is designed to be used with a CC algorithm like *ICA* or *Gibbs*. The

Local Classifier (or CC algorithm)	**Parameter set by PLUL (per relational feature)**	**Values tested by PLUL (default in bold)**
Naive Bayes (NB)	Hyperparameter α for Dirichlet prior	**1**, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096
Logistic Regression (LR)	Variance σ^2 of Gaussian prior	5, **10**, 20, 40, 80, 160, 320, 640, 1280, 2560, 512
k-Nearest Neighbor (kNN)	Weight w_R	0.01, 0.03, 0.0625, 0.125, 0.25, 0.5, 0.75, **1.0**, 2.0
LBP	Variance σ^2 of Gaussian prior	5, **10**, 20, 100, 200, 1000, 10000, 100000, 1000000

Table 2: The classifiers (NB, LR, and kNN) and CC algorithm (*LBP*) used in our experiments for which PLUL can be applied to improve performance. The second column lists the key relational parameters that we identified for PLUL to learn, while the last column shows the values that PLUL considers in its cross-validation.

last row demonstrates how it can instead be applied to a global algorithm like *LBP*. For instance, for the NB classifier, most previous research has used either no prior or a simple Laplacian ("add one") prior for each conditional probability. By instead using a Dirichlet prior (Heckerman, 1999), we can adjust the "hyperparameter" α of the prior for each relational feature. Larger values of α translate to less extreme conditional probabilities, thus tempering the impact of relational features. For the kNN classifier, reducing the weight of relational features has a similar net effect. For the LR classifier and the *LBP* algorithm, both techniques involve iterative MAP estimation. Increasing the value of the variance of the Gaussian prior for relational features causes the corresponding parameter to "fit" less closely to the training data, again making the algorithm more cautious in its use of such relational features.

While the core mechanism of PLUL—cross-validation tuning—is common, techniques like PLUL to explicitly compensate for the bias incurred from training on a fully-labeled set while testing using estimated labels have not been previously used for CC. A possible exception is Lu and Getoor (2003a), who appear to have used a similar technique to tune a relational parameter, but, in contrast to this work, they did not discuss its need, the specific procedure, or the performance impact.

PLUL attempts to compensate for the bias incurred from training on the correctly-labeled training set. Alternatively, Kou and Cohen (2007) describe a "stacked model" that learns based on estimated, rather than true labels. While the original goal of this stacked approach was to produce a more time-efficient algorithm, Fast and Jensen (2008) recently demonstrated that this technique, by eliminating the bias between training and testing, does indeed reduce "inference bias." This reduced bias enables the stacked models to perform comparably to Gibbs sampling, even though the stacked model is a simpler, non-iterative algorithm that consequently has higher learning bias. Interestingly, Fast and Jensen (2008) note that the stacked model performs an "implicit weighting of local and relational features," as with PLUL. The stacked model accomplishes this by varying the learning and inference procedure, whereas PLUL modifies only the learning procedure, and thus works with any inference algorithm that relies on a learned model.

5.6 Computational Complexity and the Cost of Caution

For learning and inference, all of the CC algorithms (variants of *ICA*, *Gibbs*, *wvRN*, and *LBP*) use space that is linear in the number of nodes/instances (N_I). *ICA* and *Gibbs* have significant similarities, so we consider their time complexity first. For these two algorithms, the dominant computation costs for inference stem from the time to compute relational features and the time to classify each node with the local classifier. Typically, nodes are connected to a small number of other instances, so the first cost is $O(N_I)$ per iteration. For the second cost, the time per iteration is $O(N_I)$ for NB and LR, and $O(N_I^2)$ for kNN. However, the number of iterations varies significantly. Based on previous work (Neville and Jensen, 2000; McDowell et al., 2007a), we set $n = 10$ for variants of *ICA*; more iterations did not improve performance. In contrast, *Gibbs* typically requires *thousands* of iterations.

Adding or removing cautious inference to *ICA* and *Gibbs* does not significantly change their time complexity. In particular, $Gibbs_{NC}$ has the same complexity as *Gibbs*. ICA_C introduces an additional cost, compared to *ICA*, of $O(N_I log N_I)$ per iteration to sort the nodes by confidence. However, in practice classification time usually dominates. Therefore, the overall computational cost per iteration for all variants of *ICA* and *Gibbs* are roughly the same, but the larger number of iterations for variants of *Gibbs* makes them much more time-expensive than *ICA*, ICA_{Kn}, or ICA_C.

LBP does not explicitly compute relational features, but its main loop iterates over all neighbors of each node, thus again yielding a cost of $O(N_I)$ per iteration under the same assumptions as above. We found that *LBP* inference was comparable in cost to that of *ICA*, which agrees with Sen and Getoor (2007). However, training the *LBP* classifier is much more expensive than training the other algorithms. *ICA* and *Gibbs* only require training the local classifier, which involves zero to one passes over the data for kNN and NB, and a relatively simple optimization for LR. On the other hand, training *LBP* with conjugate gradient requires executing *LBP* inference many times. We found this training to be at least an order of magnitude slower than the other algorithms, as also reported by Sen and Getoor (2007). LBP_{NC} has the same theoretical and practical time results as *LBP*.

wvRN is the simplest CC algorithm, since it requires no feature computation and the key step of each iteration is a simple average over the neighbors of each node. As with previous algorithms, assuming a small number of neighbors for each node yields a total time per iteration of $O(N_I)$. Prior work (Macskassy and Provost, 2007) suggested using a somewhat larger number of iterations (100) than with *ICA*. Nonetheless, in practice *wvRN*'s simplicity makes it the fastest algorithm.

Finally, all of the algorithms, except for *wvRN*, can be augmented with cautious learning via PLUL. Executing PLUL requires repeatedly running the CC algorithm with different values of the selected parameter. We used 9-13 different parameter values, and hence the cost of PLUL vs. not using PLUL is about one order of magnitude.

6. Evaluation Methodology

This section describes our hypotheses and the method that we use to evaluate them.

6.1 Hypotheses

Table 3 summarizes our five hypotheses. As described in Section 1, we expect cautious behaviors to be more important when there is a higher probability of incorrect relational inference. Thus, each

Data characteristic	**Type of caution considered**	**Hypothesis:** relative gain of caution will increase as value of characteristic...
Autocorrelation	Inference	...increases (H1)
Attribute predictiveness	Inference	...decreases (H2)
Link density	Inference	...decreases (H3)
Labeled proportion	Inference	...decreases (H4)
Labeled proportion	Learning	...decreases (H5)

Table 3: The five hypotheses that we investigate.

hypothesis varies one data characteristic that impacts the likelihood of such errors. In particular, hypotheses H1-H4 vary a data characteristic to measure the impact of cautious inference, which Section 7 will evaluate for different pairs of cautious and non-cautious inference algorithms. We define the "relative gain of cautious inference" as the difference between the accuracies of two such algorithms (e.g., *Gibbs* vs. $Gibbs_{NC}$). Hypothesis H5 also varies a data characteristic, but does so to measure the "relative gain of cautious learning" (i.e., comparing performance with vs. without PLUL).

- **H1: The relative gain of cautious inference increases with increasing autocorrelation**. Larger autocorrelation implies that relations are more predictive, and will be learned as such by the classifier. This magnifies the impact that an error in a predicted label can have on linked nodes. Therefore, we expect cautious inference algorithms to improve classification by a greater margin in such cases.

- **H2: The relative gain of cautious inference increases with decreasing attribute predictiveness (*ap*)**. Decreased *ap* implies a greater potential of errors/uncertainty in the predicted labels. The effect of cautiously using uncertain labels should be greater in such cases.

- **H3: The relative gain of cautious inference increases with decreasing link density (*ld*).** When the number of links is high, a single mispredicted label has relatively little impact on its neighbors. As the number of links decreases, however, a single misprediction can cause larger relational feature uncertainty, increasing the need for caution.

- **H4: The relative gain of cautious inference increases with decreasing labeled proportion (*lp*).** When *lp* is high, only a few of each node's neighbors have estimated labels (most are known with certainty). Consequently, there is less uncertainty in relational feature values, and less need to use estimated labels cautiously.

- **H5: The relative gain of cautious learning with PLUL increases with decreasing labeled proportion(*lp*).** As with H4, when *lp* is high there is less uncertainty in the relational features. Thus there is less disparity between the fully correct training set (where classifier parameters were learned) and the test set. Consequently, we expect PLUL, which compensates for any such disparity, to matter less when *lp* is high.

6.2 Tasks

We will evaluate three general tasks (see Section 2.3):

Parameter	Abbrev.	Values tested (defaults in bold)
Nodes per graph	N_I	250
Number of class labels	N_C	5
Number of attributes	N_A	10
Degree of homophily	*dh*	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, **0.7**, 0.8, 0.9
Link density	*ld*	0.1, **0.2**, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
Attribute predictiveness	*ap*	0.1, 0.2, 0.3, 0.4, 0.5, **0.6**, 0.7, 0.8, 0.9
Labeled proportion	*lp*	**0%**, **10%**, 20%, 40%, **50%**, 60%, 80%

Table 4: Synthetic data parameters. Defaults were chosen based on averages from Cora and Citeseer, two commonly studied data sets for CC.

1. **Out-of-sample task:** Here the test set does not contain or link to any known nodes, as with Neville and Jensen (2000), Taskar et al. (2002), and Sen and Getoor (2006).

2. **Sparse in-sample task:** Here some of the test nodes, but only a few, have known labels (we use 10%). We focus particularly on this task, because some researchers argue that it is the most realistic scenario, since often networks are large, and acquiring known labels is expensive (Bilgic and Getoor, 2008). This was the primary scenario considered by the recent work of McDowell et al. (2007a,b), Bilgic and Getoor (2008), and Gallagher et al. (2008).

3. **Dense in-sample task:** Here a substantial number of test nodes may have known labels (we use 50%). This task was the one recently evaluated by Sen et al. (2008).

6.3 Data

We evaluate the hypotheses over both synthetic and real-world data sets, which we describe below. We use the synthetic data to highlight how different data characteristics affect the relative gain of cautious behaviors, then the real-world data sets to validate these findings.

6.3.1 SYNTHETIC DATA

We use a synthetic data generator (see Table 4) with two components: a Graph Generator and an Attribute Generator. The Graph Generator has four inputs: N_I (the number of nodes/instances), N_C (the number of classes), *ld* (the link density), and *dh* (the degree of homophily). For each link, *dh* controls the probability that the linked nodes have the same class label; higher values yield higher autocorrelation (see Appendix A for details). The final number of links is approximately $N_I/(1-ld)$, and the final link degrees follow a power law distribution, which is common in real networks (Bollobás et al., 2003). The Graph Generator is identical to that used by Sen et al. (2008); see that article for more detail.

To make this a practical study, we chose default parameter values that mimic characteristics of two frequently studied CC data sets, Cora and Citeseer (McDowell et al., 2007a; Neville and Jensen, 2007; Sen et al., 2008). In particular, N_C=5 classes and Table 4 shows additional default values. We chose N_I=250 nodes, a smaller value than with Cora/Citeseer, to reduce CC execution time, but larger values did not change the performance trends.

The Attribute Generator generates 10 (N_A) binary attributes. Our design for it is motivated by our observations of common CC data sets. We found that, unlike synthetic models used in prior studies, different attributes vary in their utility for class prediction. To simulate this, we associate each attribute h with a particular class c_m, where $m = h \ mod \ N_C$, and vary the strength of each attribute's predictiveness based on the value of h. In particular, for node v_i with class y_i, the probability that v_i's h^{th} attribute x_{ih} has value 1 depends upon the class y_i as follows:

$$P(x_{ih} = 1 | y_i = c_k) = \begin{cases} 0.15 + (ap - 0.15) \cdot \frac{h}{N_A - 1} & \text{if } k = h \ mod \ N_C \\ 0.1 & \text{if } k = (h-1) \ mod \ N_C \\ 0.05 & \text{if } k = (h+1) \ mod \ N_C \\ 0.02 & \text{otherwise.} \end{cases}$$

The first line indicates that, when y_i $(= c_k)$ is the class associated with attribute h (i.e., $k = h \ mod \ N_C$), then $P(x_{ih} = 1 | y_i = c_k)$ ranges from 0.15 for $h = 0$ to ap (a constant representing the strength of attribute predictiveness) for $h = 9$. As a result, each of the five classes has two attributes associated with that class, but some classes have associated attributes that are more useful for prediction. However, x_{ih} may also be 1 when y_i is some other class besides an "associated class"; the next three lines encode this class ambiguity. This ambiguity/noise is based on our observations of Cora and Citeseer and is similar to the binomial distribution used by Sen et al. (2008).

Finally, we use a parameter for test set generation called lp (labeled proportion), which is the proportion of test nodes with known labels. We use default values of lp=0%, lp=10%, and lp=50% for the three tasks defined in Section 6.2. Nodes to be labeled are selected uniformly at random from the test set until the desired value of lp is reached. In contrast, some real data sets are likely to exhibit non-uniform clustering of known nodes. We conjecture that such data sets will have a smaller "effective" lp, since each known node will have, on average, fewer direct connections to unknown nodes. For instance, a data set with lp=10% may behave more like a data set with lp=5% where the labels are more uniformly distributed. Such effects should be examined in future work.

6.3.2 "REAL-WORLD" DATA SETS

We consider the following five "real-world" data sets (see Table 5). "Real-world" is a somewhat subjective term; however, all of the data sets are based on naturally arising networks and have been used in some form for previous research on relational learning.

1. **Cora (McCallum et al., 2000b):** A collection of machine learning publications categorized into seven classes. The relational links are (directed) citations.

2. **Citeseer (Lu and Getoor, 2003a):** A collection of research publications drawn from CiteSeer. The relational links are (directed) citations.

3. **WebKB (Craven et al., 1998):** A collection of web pages from four computer science departments categorized into six classes (Faculty, Student, Staff, Course, ResearchProject, or Other). "Other" is problematic because it is too general, representing 74% of the pages. Like Taskar et al. (2002), we discarded all "Other" pages that did not have at least three outgoing links, yielding a total of 1541 instances of which 30% are Other. The relational links are the (directed) hyperlinks among these pages.

	Cora	CiteSeer	WebKB	HepTH	Terror
Characteristics of entire graph					
Instances/nodes	2708	3312	1541	2194	645
Attributes (non-relat. feats.) available	1433	3703	100	387	106
Attributes used (max)	100	100	100	100	100
Attributes used (default)	20	20	40	40	2
Link/relation directedness	directed	directed	directed	directed	undirected
Type of relational features used	in,out	in,out	in,out,co	in,out	linksto
Class labels	7	6	6	7	6
Total relational features used	14	12	18	14	6
Links per node	3.9	2.7	5.8(64.6)	8.9	9.8
Autocorrelation	0.88	0.83	0.30(0.53)	0.54	0.16
Characteristics of each test set (on average)					
Instances/nodes	400	400	335-469	300	150
Number of folds	5	5	4	5	3
Links per node	2.7	2.7	5.7(61.0)	4.3	12.3
Approx. link density	0.23	0.23	0.64(0.97)	0.53	0.79
Autocorrelation	0.85	0.84	0.38(0.53)	0.64	0.24
Label consistency	0.78	0.75	0.21(0.90)	0.61	0.56
Approximate homophily	0.74	0.70	0.05(0.88)	0.54	0.47

Table 5: Summary of the five real-world data sets used. *in* and *out* features compute separate values based on incoming or outgoing links, while *linksto* features make no such distinction. *co* features are based on virtual co-citation links; nodes A and B are linked via a *co* link if there exists some node C with outgoing links to both A and B. For WebKB, the first statistic listed is computed ignoring co-links, while the statistic in parentheses is computed using *only* co-links. Label consistency is the percentage of links connecting nodes with the same label; Appendix A defines this and approximate homophily. Section 6.9 describes the "default" number of attributes used.

4. **HepTH:** A collection of journal articles in the field of theoretical high-energy physics, derived from the Proximity Hep-Th database (http://kdl.cs.umass.edu/data/hepth). The original data set did not have any single class label, but some pages were classified into topic subtypes. Among pages with one such subtype, we selected all articles belonging to the six most common subtypes, yielding 1404 articles. To create a more connected graph, we also selected all articles with a date after 2001 that linked to at least two of the 1404 pre-selected articles. There were 790 such articles, which we treated as having a class label of "Other." The relational links are the (directed) citations among all 2194 articles.

5. **Terror (Zhao et al., 2006):** A collection of terrorist incidents, drawn from the Profile in Terror project (http://profilesinterror.mindswap.org). The incidents are non-uniformly distributed into six categories: Bombing (44%), WeaponAttack (38%), Kidnapping (14%), Arson (2%), NBCRAttack (1%), and OtherAttack (1%). The relational links indicate (undirected) geographical co-location.

These data sets are intended to demonstrate CC performance on a range of data characteristics. For instance, CC would be expected to be very helpful for Cora and CiteSeer, where autocorrelation is high, but not very helpful for Terror.

6.4 CC Algorithms

We evaluate the ten algorithms listed in Table 1, plus *Content Only* (*CO*), a non-relational baseline that uses only attributes. For each of the four main sections in Table 1, there is one non-cautious variant (*ICA*, $Gibbs_{NC}$, LBP_{NC}, and $wvRN_{ICA+NC}$) and one or two cautious variants (ICA_C, ICA_{Kn}, *Gibbs*, *LBP*, $wvRN_{RL}$, and $wvRN_{ICA+C}$). The *wvRN* algorithms also serve as a collective, relational-only baseline.

Based on previous work (Neville and Jensen, 2000; McDowell et al., 2007a), the *ICA*-based algorithms used $n = 10$ iterations; more iterations did not improve performance. For *Gibbs*, we used 1500 iterations, with a random restart every 300 iterations, and ignored the first 100 iterations after a restart for burn-in. Additional iterations did not improve performance. $Gibbs_{NC}$ converged in far fewer iterations because it does not sample and is deterministic; we used $n = 50$.

For *LBP*, we assumed that each parameter was a priori independent and had a zero-mean Gaussian prior with a default uniform prior variance of $\sigma^2 = 10$, which is similar to the values reported in previous work (e.g., Sen and Getoor 2006; Neville and Jensen 2007). We used MAP estimation to estimate these parameters based on conjugate gradient. σ^2 controls how tightly the parameters fit to the training data; Table 2 shows the alternative values of σ^2 considered by PLUL to constrain this fitting for the relational parameters.

6.5 Classifiers

To account for possible variations in overall CC performance trends due to the effect of the underlying classifier, we tested three local classifiers with each CC algorithm wherever applicable (this excludes *LBP* and *wvRN*). Section 5.5 already described, for each classifier, the key relational feature whose value is learned by PLUL; we now provide more detail on each classifier and its application of PLUL.

The first classifier is Naive Bayes (NB). PLUL was used to learn α for the Dirichlet prior of each relational feature. The second classifier is Logistic Regression (LR). We used MAP estimation with Gaussian priors to learn the parameters for LR; PLUL learned an appropriate variance σ^2 for the prior of each relational feature. The final classifier is k-Nearest Neighbor (kNN); we used k=11. When computing similarity, attributes were assigned a weight of 1. PLUL learned the weight w_R for each relational feature. Weighted similarity was used for voting.

For each classifier, Table 2 shows the specific values considered by PLUL. The "default" value shown (e.g., $\alpha = 1.0$ for NB) was used in two ways. First, the default was used as the parameter value for all attributes. Second, the default was used for a manual setting for the parameter value for all relational features when PLUL is not being used. When PLUL was used, the learned value was used instead for the relational features.

The ICA_C algorithm requires a classifier that can ignore *missing* relational feature values. kNN and NB can do this easily: kNN by dropping the feature from the similarity calculation and NB by skipping the feature in probability computation. For LR, however, dealing with missing values is a current research topic (e.g., Fung and Wrobel 1989), with typical techniques including mean value substitution or multiple imputation. However, for CC the situation is less complex than the more general case, because missing values occur only for the test set, only for relational features, and typically only when all neighbors of a node have missing labels. Thus, we can learn several LR classifiers: one that uses all relational features, and one for each combination of features that may be missing simultaneously (for our data, this is at most 4). Experimentally, we found this

to perform better than mean value substitution, though the difference was slight because missing values were rare. These results are consistent with those of Saar-Tsechansky and Provost (2007) on non-relational data. Section 7.8 discusses this effect in more detail.

6.6 Node Representation

Each node is represented by a set of (non-relational) attributes and relational features. Algorithms based on *LBP* and *wvRN* reason directly with each individual link, and their algorithms thus directly define the effective relational features used. Approaches based on *ICA* and *Gibbs*, however, use some kind of aggregation function to compute their relational feature values. We first describe the possible aggregation functions for these features, then separately describe the complete representation for the synthetic and real data.

6.6.1 RELATIONAL FEATURES CONSIDERED

We considered three different types of relational features:

- **Count**: This type represents the number of neighbors that belong to a particular class. For each node i, there is one such feature $f_i(c)$ per class label c. The value of $f_i(c) = Neighbors_i(c)$, which is the number of nodes linked to node i that have a known or current estimated label of c. For instance, in step C of Figure 1, $f_2(P) = 1$ and $f_2(S) = 2$.

- **Proportion**: This feature is like "count", except that the feature value represents the proportion of neighbors that have a particular label, rather than the raw number of such neighbors. For this feature, $f_i(c) = Neighbors_i(c)/Neighbors_i(*)$, where $Neighbors_i(*)$ is the number of nodes linked to node i that have any current label (known or estimated, but, for ICA_C, excluding those nodes whose label was set to *missing* because of low confidence). If $Neighbors_i(*)$ is zero, then $f_i(c)$ is set to *missing*. For example, if proportion features were being used, then the feature values for step C of Figure 1 would be $f_2(P) = 1/3$ and $f_2(S) = 2/3$.

- **Multiset**: Proportion and count features aggregate the labels of a node's neighborhood to produce a single numerical value for each possible label. During inference, this aggregate value is then compared against the mean value from the training set (with NB or LR), or compared against the aggregate values for nodes in the training set (with kNN). In contrast, a "multiset" feature uses a single multiset to represent the current labels of a node's neighbors. For instance, if multiset features were used, then for step C of Figure 1, $f_2 = \{P,S,S\}$. This has the same information content as with count features, but can be exploited differently by some local classifiers. In particular, during NB inference, each label in the multiset (excluding *missing* labels) is separately used to update the conditional probability that a node has true label c. This is the "independent value" approach introduced by Neville et al. (2003b) and used by Neville and Jensen (2007). However, this approach does not directly apply to LR or kNN.

6.6.2 SYNTHETIC DATA NODE REPRESENTATION

Each node is represented by ten binary attributes and some relational features. Because representation choices can affect how well a CC algorithm handles the uncertainty of estimated labels, for each local classifier-based algorithm we considered count and proportion relational features, as well

as multiset features when using NB. For each trial, we evaluated the two or three possible types of relational features with cross-validation (evaluating accuracy on the holdout set), then selected the feature type with the highest accuracy to use for testing. When PLUL was used, PLUL was also applied to each feature type; the best performance (on the holdout set) reported by PLUL for each feature type was then used for this feature selection. Section 7.8 describes which feature types were chosen most often for each local classifier. Since there are 5 class labels for the synthetic data and links are undirected, there were 5 relational features when using count or proportion features, and 1 relational feature (whose value is a multiset) when using multiset.

6.6.3 REAL-WORLD DATA NODE REPRESENTATION

For all five data sets we used binary attributes that indicated the presence or absence of a particular word. For WebKB, these words were from the body of each HTML page; we selected the 100 most frequent such words, which was all that was available in our version of the data set. For symmetry, and because adding more words had a small impact on performance, we likewise set up the remaining data sets to select 100 words as attributes. For Cora and CiteSeer, these words were taken from the body of the publications; as with previous work (McDowell et al., 2007a) we selected the 100 words with the highest information gain in the training set to use. For Terror, the words come from hand-written descriptions of each incident provided with the data set; we selected the first 100 of the 106 available attributes. For HepTH, we selected, based on information gain, the 100 highest-scoring words from the article title or the name of the corresponding journal.

For relational features, we again considered the proportion, multiset, and count features, and selected the best feature type using cross-validation as described above. All of the data sets except Terror had directed links. For these data sets, we computed separate relational feature values based on incoming and outgoing links. In addition, previous work has shown WebKB to have much stronger autocorrelation based on co-citation links than on direct links (see Table 5). However, using such links can sometimes be problematic. Thus, we evaluate two data sets: "WebKB" and "WebKB+co". For WebKB, algorithms use in and out links ("direct" links). For WebKB+co, algorithms use in, out, and co-links, except *wvRN* uses only co-links, as suggested by Macskassy and Provost (2007) (see Section 7.6).

6.7 Training/Test Splits Generation

For the synthetic data, we generate training, holdout, and test graphs that are disjoint. Likewise, for WebKB, the data was already divided into four splits (one for each department) that can be used for cross-validation.

For the other real data sets, we must manually construct training and test splits from the original graph. Sen et al. (2008) suggest a technique based on snowball sampling that involves picking a random starting node and iteratively growing a split around that node, where the class of the next node to be selected is sampled from the overall class distribution. However, we found that low graph connectivity often prevented the algorithm from producing a final subgraph whose class distribution resembled the whole graph's. Instead, we created the following technique, *similarity-driven snowball sampling*: given the whole graph G, pick a random starting node and add it to the split G_1. At each step, consider the *frontier* F of G_1 (all those nodes not in G_1 that link to some node in G_1). Among all labels c that exist in F, select the class label c' such that adding some node of label c' to G_1 would maximize the similarity (inverse Euclidean distance) of the class distributions

of G_1 and G. Given this c', randomly select some node in F of class c' and add it to G_1. Repeat this random selection and insertion until G_1 is of the desired size.

We run this algorithm in parallel for N_S different subgraphs, using N_S different seeds, and permit each node to be inserted into only one subgraph. This results in N_S disjoint splits that have similar class distributions and that can be used for N_S-fold cross validation. We set $N_S = 5$ for Cora, Citeseer, and HepTH, and $N_S = 3$ for the smaller Terror.

Table 5 shows some of the characteristics of the generated test sets vs. the original, complete graphs. In general, the autocorrelation and number of links per node are similar, indicating that the sampling procedure did not dramatically change the average characteristics of the graph. While the splitting procedure effectively removes links, the average degree of the test sets may still be greater than with the original graph if high-degree subsets of the original are selected.

6.8 Test Procedure

We first consider the synthetic data. For each control condition (i.e., data generated with a combination of *dh*, *ap*, *ld*, and *lp* values, see Table 4) we ran 25 random trials. For each trial, we generated training, holdout, and test data sets of 250 nodes each. All training is performed on the fully labeled training set. The holdout set, when not used for PLUL, was merged with the training set. We measured classification accuracy on the test set, excluding all nodes with "known" labels.

For the real-world data sets, each experiment involves using all of the relational features shown in Table 5 and a fixed number of attributes (N_A). We vary N_A from 2 to 100 (recall that for all data sets 100 attributes were selected for experimentation). For each setting of N_A, we perform N_S-fold cross-validation, where N_S is 3, 4, or 5, depending on the data set. Each one of these 3 to 5 trials is associated with one subgraph (the test set), and the remaining 2-4 subgraphs comprise the training set. We then apply PLUL by training on half of the training set and using the other half as the holdout set. After PLUL selects the best parameter setting, we re-train on the whole training set and evaluate accuracy on the test set. If PLUL is not used, training likewise uses the whole training set.

We report results with accuracy in order to ease comprehension of the results and to facilitate comparison with some of the most relevant related work (e.g., Sen et al., 2008; Macskassy and Provost, 2007). Results with area under the ROC curve (AUC) for the majority class demonstrated similar trends.

6.9 Statistical Analysis

We conducted two distinct types of analysis. First, to compare algorithms for a single control condition, we used a one-tailed paired t-test accepted at the 95% confidence level. For every such test each "test point" is the accuracy over a single trial's test graph. For example, for the synthetic data there are 25 trials for each control condition, and thus a single t-test compares 25 pairs of accuracies (e.g., ICA_C vs. ICA). In all cases the test graphs used by these t-tests are disjoint, for both the synthetic and the real data.

Second, we performed linear regression slope tests. In particular, for hypotheses H1-H4, we compared two algorithms (e.g., ICA_C vs. ICA) for each independent variable X (e.g., *ld*) as follows: For each trial, we computed the difference in the algorithms' classification accuracies (e.g., for the synthetic data, 225 such differences for 25 trials and 9 values of *ld*). We performed linear regression ($Y = a + bX$), where the accuracy difference is the dependent variable Y and X is the independent variable (e.g., *ld*). The estimated value of slope b, when non-zero, indicates an increasing (+)

or decreasing ($-$) trend. Regression produces a p value associated with the slope that indicates the significance level for hypothesis testing; we accept when $p < 0.05$. For hypothesis H5, the equations are the same but we compare a single CC algorithm with and without PLUL.

For the synthetic data, the analysis is straight-forward and we use the data generation parameters *dh*, *ap*, *ld*, and *lp* as the independent variable for regression. Analysis for the real data sets requires more explanation. For instance, each computed subgraph of a data set has similar autocorrelation, so regression for H1 (where autocorrelation is the X value) cannot be performed on a single data set. Instead, we combine the trials of all the real data sets into one analysis, where the independent variable is the measured autocorrelation of the corresponding data set (we include WebKB, but exclude WebKB+co because it's not clear how to compute its autocorrelation with direct links combined with co-citation links). In addition, our results show that when attribute predictiveness is high, there is less need for caution. Thus, to prevent any interactions between autocorrelation and caution from being obscured by high attribute predictiveness, we use fewer than 100 attributes for these experiments. In particular, for each data set we evaluated the baseline *CO* algorithm with varying numbers of attributes N_A, and selected the number that yields an average accuracy closest to 50%. Table 5 shows the resulting default number of attributes for each data set.

For H2 (attribute predictiveness), we can directly vary the number of attributes, so we can perform regression for each data set separately. However, attribute predictiveness is typically not a linear function of the number of attributes. Thus, for H2 we perform regression where the dependent variable is the accuracy of *CO* for each trial (as a surrogate for attribute predictiveness).

We do not directly evaluate H3 for the real data sets (see Section 7).

For H4 and H5 (varying labeled proportion), we directly vary *lp*, so we can compute separate results for each data set. Moreover, *lp* is suitable for direct use as the dependent variable. As with H1, we use the default number of attributes for each data set in order to avoid having high attribute predictiveness obscure the interaction of caution and *lp*. We omit nonsensical points (e.g., *wvRN* when *lp*=0%) from all of the analyses.

Finally, for each hypothesis we also perform a *pooled analysis*. For the synthetic data, this involves pooling the results of all the cautious CC algorithms, then performing the slope regression test. For the real-world data, we pool the results across both the CC algorithms and each of the real data sets. In addition, to account for differences in the data sets, we perform a multiple regression analysis that includes autocorrelation as one of the input variables (except for H1). In particular, we fit the data to the line $Y = a + b_1X_1 + b_2X_2$, where X_1 is the variable in question (e.g., *lp* for H4 or H5) and X_2 is the autocorrelation of the data set. The X_2 term factors out differences due only to autocorrelation, thus making the other trends more clear. The p-value corresponding to b_1 is then used for hypothesis testing.

6.10 Implementation Validation

To validate the implementation of our algorithms, we replicated three different synthetic data generators: those used by Sen and Getoor (2006), Neville and Jensen (2007), and Sen et al. (2008). We then replicated some of the experiments from these papers. While several of our CC algorithm variants were not evaluated in any of these earlier papers, we were able to compare results for *ICA*, *Gibbs*, and *LBP*, with the LR and NB classifiers as appropriate, and found very consistent results. Section 8.4 discusses one exception.

LBP is the most challenging algorithm to implement and to get to converge. To deal with such problems, Sen et al. (2008) seeded *LBP*'s learning process with weights learned from *ICA*. Alternatively, we found that seeding with values estimated from empirical counts over the data, combined with limiting the maximum step size of the search to prevent oscillation, worked well. With these enhancements, *LBP* achieved equivalent accuracy to that reported by Sen and Getoor (2006), and, when PLUL was applied, significantly improved it for the cases of high homophily and link density (where *LBP*'s accuracy had been very poor). In contrast, we found that *LBP* could replicate the performance of Sen et al. (2008), but that in this case PLUL had little effect. Section 8.4 explains the data characteristics of that study (effectively high *lp*) that led to this result.

7. Evaluation Results

This section presents our experimental results. Section 7.1 presents a summary of the results, Section 7.2 explains how we present the detailed results, and subsequent sections discuss these detailed results for each hypothesis. We focus on the sparse in-sample task, so *we accept a hypothesis if it is confirmed, for the* lp=*10% case, by the pooled analysis on both the synthetic data and the real-world data*. Hypotheses H4-H5 involve varying *lp*; here we accept the hypothesis if confirmed on both the synthetic and real data.

When a local classifier is needed, all results below use NB by default. We found that NB's performance was better or equivalent to that of LR and kNN in almost every case (see Section 8.4), for both the synthetic and real data sets, and that using LR or kNN led to very similar performance trends. Below we mention some of the results for LR and kNN; see the online appendix for more detail. In addition, PLUL is used everywhere unless otherwise specified; see analysis and motivation in Section 7.7.

7.1 Summary of Results

Tables 6-8 summarize our overall results for hypotheses H1-H5. Each table presents results for the synthetic data on the left and (where applicable) for the real data sets on the right. Each reported value represents the estimated slope of the line measuring the difference between a cautious and a non-cautious CC algorithm as the corresponding x-parameter (e.g., autocorrelation) is varied (see Section 6.9). Only values that were statistically different from zero are reported; otherwise a dash is shown. Bold values indicate a significant slope that supports the corresponding hypothesis. For instance, H2 predicted that caution becomes more important as attribute predictiveness *decreases* (a negative slope). Thus, Table 7 shows a minus sign for the expected slope and all significant, negative slopes are shown in bold. Where possible, we show separate results for the out-of-sample, sparse in-sample, and dense in-sample tasks (using $lp = 0\%$, 10%, and 50%). However, to simplify the table the real-world data results for H2 are shown only with lp=10%; Section 7.4 describes other results.

The tables show strong support for hypotheses H1, H2, and H4. In particular, we accept H1, H2, and H4 because the pooled analyses find significant slopes in the expected direction; non-pooled results also demonstrate consistent support. Thus, the data support the claims that each cautious inference algorithm outperforms[7] its non-cautious variant by increasing amounts when autocorrelation

7. Technically, the slope results don't by themselves show that the cautious algorithms "outperform" the non-cautious algorithms—only that the relative performance of the cautious algorithms is improving in the hypothesized direction.

	Expected slope	Syn. data lp=0%	Syn. data lp=10%	Syn. data lp=50%	Real-world data lp=0%	Real-world data lp=10%	Real-world data lp=50%
H1: auto-correlation							
ICA_C vs. *ICA*	+	**+0.13**	**+0.13**	**+0.03**	—	**+0.27**	**+0.15**
ICA_{Kn} vs. *ICA*	+	n.a.	**+0.04**	**+0.02**	n.a.	**+0.15**	**+0.13**
Gibbs vs. $Gibbs_{NC}$	+	**+0.18**	**+0.15**	**+0.03**	**+0.27**	**+0.25**	**+0.15**
LBP vs. LBP_{NC}	+	**+0.10**	**+0.08**	—	—	—	—
$wvRN_{RL}$ vs. $wvRN_{ICA+NC}$	+	n.a.	**+0.43**	—	n.a.	**+0.41**	**+0.07**
$wvRN_{ICA+C}$ vs. $wvRN_{ICA+NC}$	+	n.a.	**+0.40**	—	n.a.	**+0.67**	**+0.10**
Pooled	+	**+0.13**	**+0.21**	**+0.01**	**+0.13**	**+0.30**	**+0.11**

Table 6: Summary of results for hypothesis H1. All values shown represent a slope that is significantly different from zero; values in bold support H1. For H1, at a given lp value all data sets (except WebKB+co) are used to compute a single slope value by treating the auto-correlation of the data set as the X value. All algorithms used PLUL where applicable. "n.a." indicates that the algorithm doesn't make sense at lp=0%.

	Expected slope	Syn. data lp=0%	Syn. data lp=10%	Syn. data lp=50%	Real-world data ($lp = 10\%$) Cora	CiteSeer	HepTH	WebKB+co	WebKB	Terror
H2: attribute predictiveness										
ICA_C vs. *ICA*	-	**-0.10**	**-0.25**	**-0.12**	**-0.60**	**-0.61**	**-0.29**	—	—	—
ICA_{Kn} vs. *ICA*	-	n.a.	**-0.06**	**-0.08**	**-0.14**	—	**-0.16**	—	—	—
Gibbs vs. $Gibbs_{NC}$	-	**-0.09**	**-0.27**	**-0.14**	**-0.44**	**-0.50**	—	—	—	—
LBP vs. LBP_{NC}	-	**-0.12**	**-0.28**	**-0.05**	**-0.46**	**-0.35**	—	n.c.	**-0.29**	—
Pooled	-	**-0.10**	**-0.22**	**-0.10**	**-0.23** (over all real data and CC algs.)					
H3: link density										
ICA_C vs. *ICA*	-	**-0.08**	**-0.09**	**-0.03**						
ICA_{Kn} vs. *ICA*	-	n.a.	+0.06	**-0.02**						
Gibbs vs. $Gibbs_{NC}$	-	**-0.09**	**-0.07**	**-0.04**	(not evaluated)					
LBP vs. LBP_{NC}	-	+0.12	**-0.23**	**-0.04**						
$wvRN_{RL}$ vs. $wvRN_{ICA+NC}$	-	n.a.	**-0.18**	**-0.05**						
$wvRN_{ICA+C}$ vs. $wvRN_{ICA+NC}$	-	n.a.	0.11	**-0.03**						
Pooled	-	—	**-0.07**	**-0.04**						

Table 7: Summary of results for hypotheses H2 and H3. As before, all values shown represent a slope that is significantly different from zero; values in bold support the corresponding hypothesis. All algorithms used PLUL where applicable. "n.c." indicates where *LBP* did not converge.

is higher (H1), attribute predictiveness is lower (H2), and/or the labeled proportion is lower (H4). In addition, the data show consistent interactions among these factors. In particular, the strength of

However, the raw accuracies do show consistent performance gains for the cautious algorithms, so in this context the slope results do show the cautious algorithms outperforming the others by increasing amounts.

		Syn. data	Real-world data					
	Expected slope		Cora	CiteSeer	HepTH	WebKB+co	WebKB	Terror
H4: labeled proportion (comparing cautious vs. non-cautious algorithm)								
ICA_C vs. ICA	-	**-0.09**	**-0.11**	**-0.14**	**-0.05**	—	—-	—
ICA_{Kn} vs. ICA	-	**-0.02**	**-0.06**	—	**-0.05**	**-0.29**	—	—
$Gibbs$ vs. $Gibbs_{NC}$	-	**-0.11**	**-0.14**	**-0.13**	0.05	0.28	—	—
LBP vs. LBP_{NC}	-	**-0.05**	—	—	—	n.c.	—	—
$wvRN_{RL}$ vs. $wvRN_{ICA+NC}$	-	**-0.28**	**-0.37**	**-0.39**	**-0.18**	—	—-	—
$wvRN_{ICA+C}$ vs. $wvRN_{ICA+NC}$	-	**-0.27**	**-0.36**	**-0.32**	**-0.15**	**-0.31**	—	+0.28
Pooled	-	**-0.12**	**-0.07** (over all real data and CC algs.)					
H5: labeled proportion (comparing with PLUL vs. without PLUL)								
ICA_C	-	**-0.02**	—	—	—	—	—	—
ICA_{Kn}	-	**-0.01**	—	—	**-0.02**	—	—	—
ICA	-	—	—	—	—	**-0.18**	—	—
$Gibbs$	-	**-0.02**	—	—	**-0.04**	—	**-0.07**	—
LBP	-	**-0.03**	—	—	—	n.c.	—	—
Pooled	-	**-0.02**	**-0.01** (over all real data and CC algs.)					

Table 8: Summary of results for hypotheses H4 and H5, which both vary the labeled proportion (lp). As before, all values shown represent a slope that is significantly different from zero; values in bold support the corresponding hypothesis. For H4, all algorithms used PLUL where applicable.

the dependence (the magnitude of the slope) generally decreases as the labeled proportion increases from 10% to 50% (Section 7.4 discusses the differences between lp=0% and 10% in more detail).

Table 7 shows weaker support for H3 (cautious inference gain increases as link density decreases). H3 is supported by most of the synthetic data cases and by the pooled analysis for lp=10% and lp=50%, but the magnitude of the slopes indicates a weaker effect. Moreover, Section 7.5 examines these results more closely and proposes that a more appropriate hypothesis would state that the cautious inference gain is greatest when link density is moderate. This conclusion is also tentatively supported by a per-node degree analysis of the real data.

Table 8 also shows weaker support for H5. The synthetic data results supported H5 for every algorithm except *ICA*. In addition, for 18 of the 29 possible cases shown for the real data sets, the computed slope was negative, as predicted by H5. However, the magnitude of these slopes indicate a weaker effect than with H1, H2, or H4. This decreased magnitude, in conjunction with the smaller number of trials for the real data, leads to only 4 of those 18 slopes reaching statistical significance. Nonetheless, by combining trials across algorithms and data sets, the pooled analysis does find significant (but small) negative slopes for both the synthetic and real data, so we accept H5. This indicates, as expected, that cautious learning with PLUL is most important when lp is small; Section 7.7 also demonstrates that in this case PLUL can provide substantial performance gains.

In addition to these results for each hypothesis, regarding relative performance trends as data characteristics vary, our results also show statistically significant differences between the cautious and non-cautious algorithms for at least some of the data conditions. These differences are consistent with the accepted hypotheses. For instance, using the default synthetic data characteristics, each cautious algorithm showed a significant performance gain over its non-cautious variant, and the amount of this gain increased as autocorrelation increased, attribute predictiveness decreased, or labeled proportion decreased.

7.2 Explanation of Results Presentation

In the following sections, we present several figures that compare CC algorithmic performance. In these figures some controllable parameter is the x-axis and the y-axis is the resultant accuracy for a given algorithmic variant, averaged over all trials. For instance, Figure 8 plots accuracy vs. the degree of homophily (*dh*). Each figure compares cautious and non-cautious variants of a particular CC algorithm: *ICA*, *Gibbs*, *LBP*, or *wvRN*. In addition, for the CC algorithms that use a local classifier (*ICA* and *Gibbs*), we often include results for the non-relational algorithm *CO* for comparison.

In each section below, we use these results to describe two kinds of analysis. First, we accept or reject a hypothesis, based on the pooled regression slope test. This analysis confirms or fails to confirm that the importance of the cautious techniques *does change* in the expected direction as some data parameter varies, but does not evaluate *how important* the cautious techniques are in improving performance. To answer the latter question, we report on a second analysis that evaluates, using paired t-tests, whether the cautious techniques perform significantly better than the non-cautious alternatives (see Section 6.9).

Each figure has embedded statistical information corresponding to some of these t-tests. In particular, each non-cautious CC variant is plotted with a $\times$ marker, while cautious CC variants are plotted with a triangle (where multiple cautious variants exist, two triangle orientations are used: $\bigtriangledown$ and $\bigtriangleup$). For a particular x-value, if the plotted triangle is filled in (solid color), then that cautious variant had accuracy that was significantly different from the accuracy *of the corresponding non-cautious variant*. Hollow triangles instead indicate no significant difference. This notation does *not* directly indicate other significance comparisons (e.g., between the two cautious variants ICA_C and ICA_{Kn}); where necessary we describe such results in the text. For example, in Figure 8, the graph in the third column of the first row (*LBP* at *lp*=0%) shows that *LBP* significantly outperforms LBP_{NC} when *dh*=0.6 (note the filled triangle). However, for *dh*=0.5, *LBP*'s small gain is not statistically significant (hollow triangle).

When *lp*=0%, ICA_{Kn} is equivalent to *ICA*, so results for ICA_{Kn} are not shown. Also, *LBP* with WebKB+co did not converge due to the very high number of links, so results for that case are not considered (cf., Taskar et al., 2002).

7.3 Result 1: The Relative Gain of Cautious Inference Increases with Increasing Autocorrelation

Table 6 reports that for H1, for the sparse in-sample task (*lp*=10%), the pooled regression analyses found all significant positive values for the slope B. Thus, we accept H1. In addition, all the non-pooled analyses found significant positive values. The only exception was *LBP* on the real data sets, which had a positive, non-significant slope ($b = +0.03$).

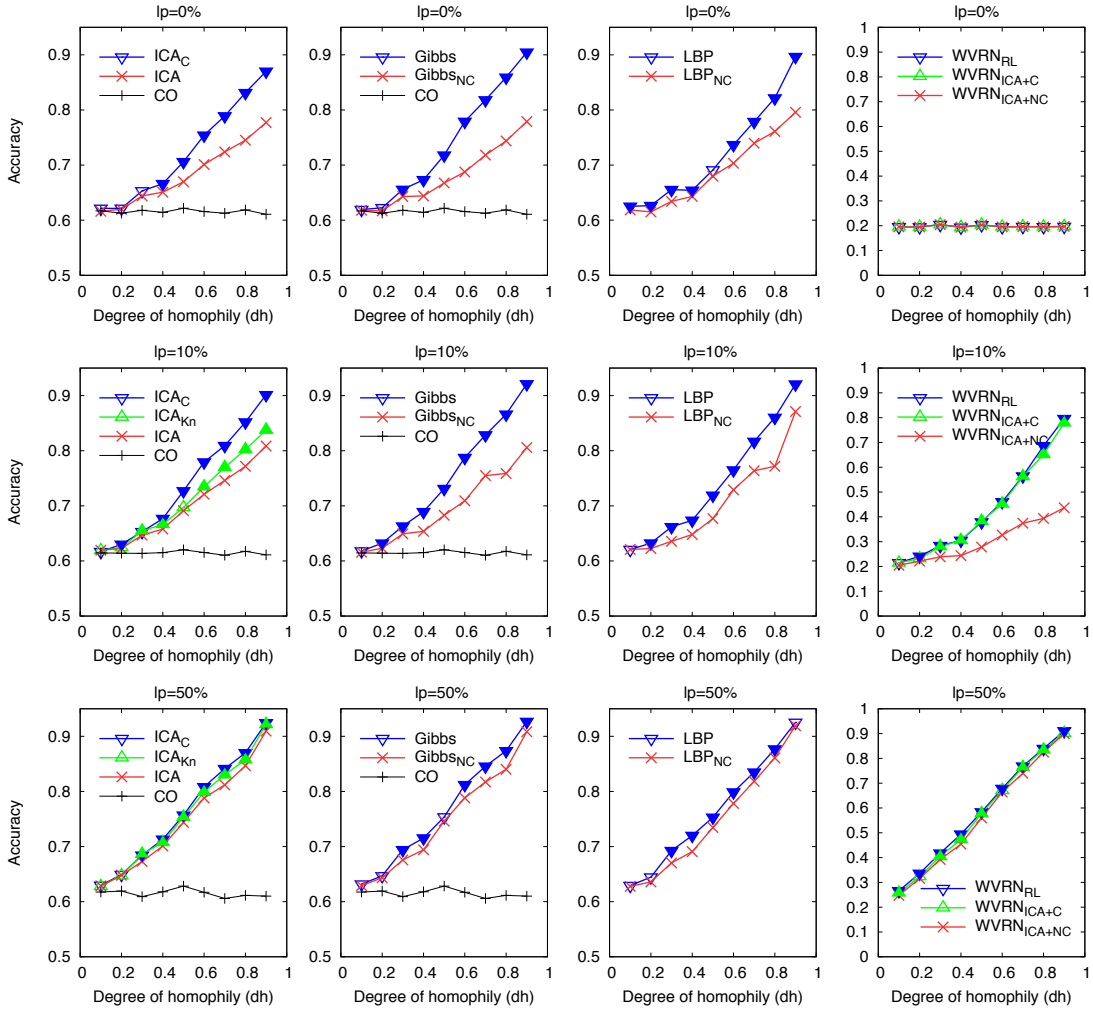


Figure 8: Results for the synthetic data as the degree of homophily (*dh*) varies. Section 7.2 explains how filled triangles indicate statistical significance. Some of the gains are small but consistent, leading to significance, as in the bottom right graph.

For *lp*=0% and *lp*=50%, the pooled analyses and most individual analyses show the same positive slopes (on the real data for ICA_C vs. *ICA* at *lp*=0%, the slope was $b = +0.11$, but the p-value was just over the significance threshold), as we also found with LR and kNN. The reduced significance and magnitude of the slopes when *lp*=50% is also consistent with our expectations, since the overall importance of caution should decrease as *lp* increases (see hypotheses H4 and H5). Section 7.4 explains more for the *lp*=0% case.

Figure 8 shows detailed performance trends for the synthetic data. Here each column presents results for different variants of a single CC algorithm (*ICA*, *Gibbs*, *LBP*, and *wvRN*), and each row shows results for a different value of *lp*. The x-axis varies homophily (which directly increases autocorrelation) and the y-axis reports average accuracy.

This figure confirms that when homophily is very low, CC offers little gain, and thus the cautious variants perform equivalently to the non-cautious variants (and, except for *wvRN*, to the non-

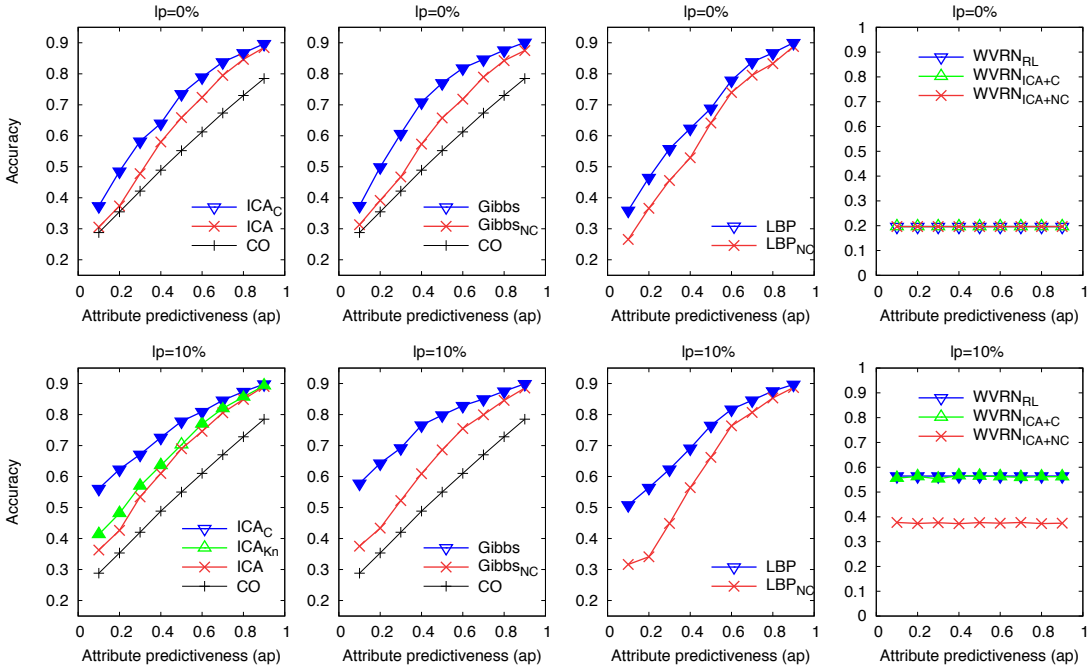


Figure 9: Results for the synthetic data as attribute predictiveness (*ap*) varies.

relational baseline *CO*). As the strength of relational influence (as well as the potential for incorrect relational inference) increases with higher homophily, the relative gain of the cautious methods increases substantially (e.g., at *lp*=10%, gains for NB-based algorithms rise from 4-5% at *dh*=0.5 to 9-12% at *dh*=0.9). The gains from caution are statistically significant in most cases when $dh \geq 0.3$. Results with LR and kNN show very similar trends (see online appendix).

Figure 8 also confirms that as *lp* increases, the cautious and non-cautious variants perform more similarly. However, even for *lp*=50%, the cautious variants maintain a significant, though smaller, advantage. In the other results discussed below, the same trend of very similar performances at *lp*=50% was evident. Likewise, the graphs for *lp*=0% are similar to those for *lp*=10%. Thus, we defer most results for *lp*=0% or 50% to the online appendix.

7.4 Result 2: The Relative Gain of Cautious Inference Increases as Attribute Predictiveness (*ap*) Decreases

Table 7 reports that, for *lp*=10%, the regression analyses found all significant negative slopes (as expected) for the synthetic data. Likewise, in almost all cases we found significant negative slopes for the real data sets that have substantial autocorrelation (Cora, Citeseer, HepTH, and WebKB+co), except for WebKB+co (which had very erratic performance with all the algorithms). We accept H2, because the pooled analysis found negative slopes at *lp*=10% for both synthetic and real data; this result also holds at *lp*=0% and 50%.

Figure 9 shows detailed performance trends for the synthetic data as the x-axis varies *ap*. For instance, for *lp*=10%, when the attribute predictiveness (*ap*) is 0.6 (the default), ICA_C and *Gibbs* outperform their non-cautious variants by 6-7%. However, as *ap* decreases to 0.2, label uncertainty

increases (as evidenced by the drop for *CO*), causing the relative gain of caution to increase to 20%. *LBP* shows very similar results.

Results for *lp*=0% are mostly similar, but with an interesting twist. In this case, the relative gain of cautious CC increases as *ap* decreases, as with *lp*=10%. However, this gain peaks at *ap*=0.2 or 0.3, then declines as *ap* continues to decrease. When attribute predictiveness is very low, and there are no known labels to help seed the inference (i.e., *lp*=0%), then even the cautious algorithms have difficulty exploiting relational information, and achieve accuracy only moderately above the baseline *CO*. However, even in this case the cautious algorithms maintain some small, statistically significant advantage over the their non-cautious variants (which at *ap*=0.1 do little better than *CO*). Also, observe that ICA_C, *Gibbs*, and *LBP* all improve substantially for the *lp*=10% case (compared to *lp*=0%), even though only ICA_C explicitly favors the provided known labels in its inference process. In this case, using caution appears to be the important performance factor, regardless of what specific behavior provides that caution.

Figures 10 and 11 provide similar results for the real data sets with *lp*=10%, where the x-axis is now the number of attributes used, which correlates with overall attribute predictiveness. In general, the trends shown are similar to those already observed for the synthetic data. In particular, the graphs for Cora, Citeseer, HepTH, and WebKB all follow the same pattern: cautious algorithms outperform non-cautious algorithms more when the number of attributes is low (and cautious ICA_C outperforms the somewhat cautious ICA_{Kn}). Consistent with H1, the magnitude of these gains varies with autocorrelation: larger for Cora and Citeseer, smaller for HepTH and WebKB, and non-existent for Terror (where autocorrelation is very weak).

There are two exceptions to the similarities of these results with the synthetic data. First, for some data sets *Gibbs* and/or *LBP* perform noticeably worse than ICA_C; we discuss this separately in Section 8.1. Second, WebKB+co shows fairly erratic performance for all algorithms except ICA_{Kn}. In general, the co-citation links used by WebKB+co appear to be very informative (peak accuracy is much higher than with WebKB), but also potentially misleading. This may be a function of the WebKB graph structure: Table 5 shows that co-citation links have a very high label consistency of 0.90 (implying that classifiers will learn a strong relational dependence), but this may be biased by the presence of some very high degree nodes. During learning the co-citation links may appear very informative on average, but this strong dependency may lead to mispredictions for low-degree nodes, leading to the observed erratic behavior.

We now briefly return to the slope analysis of Table 7. For the synthetic data, the negative slopes for H2 are significant in all cases, but generally largest for *lp*=10%. This behavior is consistent with our previously discussed analyses of the synthetic data: when *lp*=0%, the performance of cautious algorithms for very low *ap* is diminished, thus producing a smaller slope magnitude than when *lp*=10%. On the other hand, the more general observation that caution is less useful when *lp* is high explains why the magnitude of the slopes is less for *lp*=50% than for *lp*=10%. We found similar trends for the real-world data sets: while Table 7 shows significant negative slopes for H2 for most cases (excluding the erratic WebKB+co and the low autocorrelation Terror) when *lp*=10%, results (not shown) with *lp*=0% or 50% indicate slopes of reduced magnitude and/or slopes that do not reach statistical significance. However, in both cases the pooled analysis still indicates significant negative slopes for H2 (-0.05 for *lp*=0% and -0.13 for *lp*=50%).

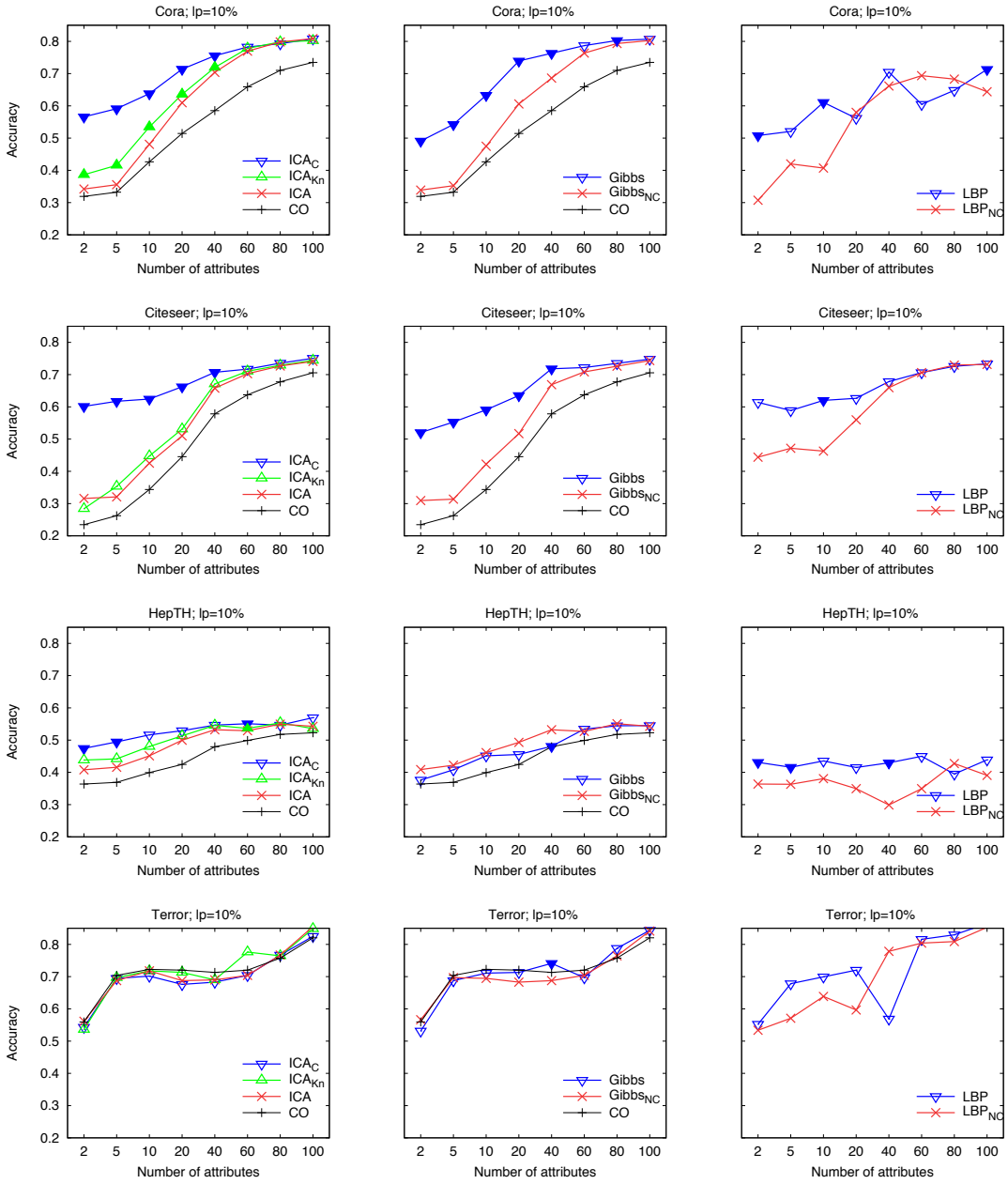


Figure 10: Results for four of the real data sets as the number of attributes is varied. The x-axis is not to scale; this is to improve readability and to yield a more linear curve for the baseline *CO* algorithm, thus facilitating comparison with Figure 9. Because there are only 3-5 trials for the real data, high variance sometimes causes substantial gains to not be statistically significant.

7.5 Result 3: The More Cautious Algorithms Outperform Non-Cautious Algorithms when Link Density (*ld*) is Moderate, But Have Mixed Results When *ld* is High

For the synthetic data, the results in Table 7 support H3 for all algorithms when *lp*=50%, for most algorithms when *lp*=10%, and for only two algorithms when *lp*=0%. The pooled analysis finds,

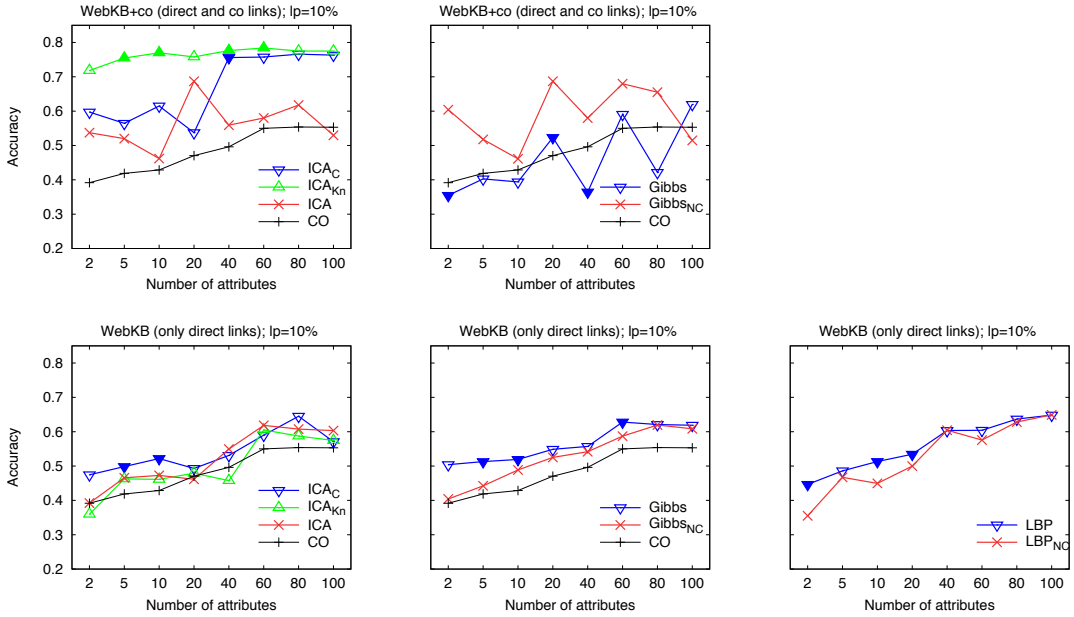


Figure 11: Results for the WebKB data sets as the number of attributes is varied. With WebKB+co, *LBP* did not converge, so results are not shown.

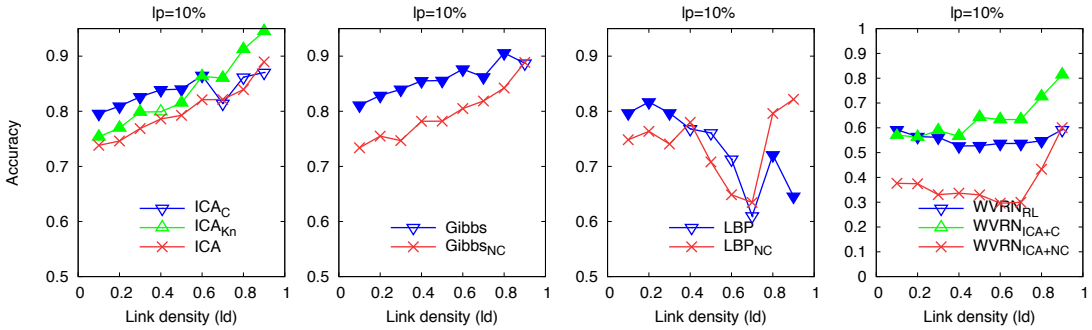


Figure 12: Results for the synthetic data as link density (*ld*) varies.

as expected, significant negative slopes for *lp*=10% and *lp*=50%. However, without corresponding pooled results for the real data, we cannot accept H3. Moreover, the results we present below will suggest a revision to H3.

Figure 12 shows the results as *ld* is varied, for *lp*=10%. When *ld* is low to moderate (up to *ld*=0.6), the cautious algorithms consistently and significantly outperform their non-cautious variants. We had hypothesized that this advantage would decrease as link density increased, because when the link graph is dense, the relational features are relatively unaffected by a few incorrect labels, and thus using such labels cautiously matter less; Figure 12 generally reflects this trend. In some cases the non-cautious algorithm even outperforms the cautious algorithm at very high *ld*. For instance, at *ld*=0.9 *ICA* outperforms the more cautious ICA_C (though not significantly). At such high link density, simply using all available information with *ICA* may work better than ICA_C's

cautious but partial use of estimated labels—provided that accuracy is high enough that errors are few. In separate experiments we confirmed that if the attribute predictiveness (and thus accuracy) was lower, ICA_C maintained it's advantage over *ICA* even when *ld* was very high.

While these results generally indicate, as expected, that the gain from caution decreases as *ld* becomes high, closer examination indicates that this gain from caution peaks not at very low *ld*, but at moderate *ld*. In particular, the gain from caution peaks when *ld* is 0.2 or 0.3 for ICA_C, ICA_{Kn}, or *Gibbs*, and when *ld* is 0.6 for $wvRN_{RL}$ and $wvRN_{ICA+C}$. In hindsight, this effect makes sense: as the number of links decrease, there is less relational influence, and thus less probability of incorrect relational influence, so caution matters less. Another effect is that with fewer links, there are fewer opportunities for a cautious algorithm to favor one node's predictions over another's.

To further analyze these trends, we turn to the real data. We did not attempt to directly vary the link density of the real data sets, because it's not clear how to realistically add links to an existing data set, as would be necessary to create a reasonable range of link densities for experimentation. However, Table 9 examines our previous results for the real data sets, showing the amount of cautious gain broken down by the link degree of each node. This approach does not directly correlate to varying the overall link density, so our conclusions are tentative, but it does provide some insight. We focus primarily on ICA_C; trends with other algorithms were similar.

The results support our previous conjectures. In particular, the cautious gain generally decreases for the highest link degrees, even going negative in some cases. Moreover, in most cases the cautious gain also decreases for the lowest link degrees, resulting in a peak for the cautious gain (shown in bold if present) at moderate link degrees. These effects generally hold true for the synthetic data and for the real data sets that have substantial autocorrelation.

We now return to Figure 12 to consider a few possible exceptions. First, with *LBP*, accuracy decreases with increasing *ld*, is erratic, and is sometimes better with LBP_{NC} than with *LBP*. This is not surprising: the short graph cycles caused by high *ld* produces great problems for *LBP* (e.g., Sen and Getoor, 2006; Sen et al., 2008). Even these *LBP* accuracies are much better than those achieved without PLUL (see Section 7.7).

Second, two of the cautious algorithms (ICA_{Kn} and $wvRN_{ICA+C}$) performed unexpectedly well, continuing to significantly outperform the non-cautious variants (and even alternative cautious variants) at very high link density. Interestingly, these effects also occur with WebKB+co (see Figures 11 and 14), which has by far the highest link density of the real data sets.[8] In addition, the superior performance of ICA_{Kn} at high *ld* remains even when the local classifier is changed to LR or kNN (see Figure 19 in the online appendix). We suspect that ICA_{Kn}'s advantage arises because it both achieves a better starting point than *ICA* (by favoring known labels in its first iteration) and exploits more information than ICA_C (by using all estimated labels in subsequent iterations—and when *ld* is high using a few erroneous labels doesn't harm performance). For $wvRN_{ICA+C}$, its advantage over $wvRN_{RL}$ must arise from the key algorithmic difference: since $wvRN_{ICA+C}$ is a hard-labeling algorithm, it gives all labeled nodes equal weight in the neighborhood average that determines the next label for a node. When link density is high, relying on this simple average may be better than $wvRN_{RL}$'s soft-labeling estimation, which implicitly gives more weight to nodes with more extreme

8. At first, these strong performances seem to conflict with Macskassy and Provost (2007), who generally find $wvRN_{RL}$ outperforms $wvRN_{ICA+C}$. However, two-thirds of their data sets are variants of WebKB, but where all "Other" pages have been removed from the classification task. This change makes the classification problem easier, and thus may explain the discrepancy. In addition, on the only other data set used in that work and this article (Cora), our performance trends are very similar.

	Degree 1-2	Degree 3-5	Degree 6-10	Degree 11-20
Synthetic data, using NB+ICA_C				
lp= 0%	5.5%	8.2 %	**13.8%**	8.2%
lp=10%	5.2%	9.7 %	8.6%	10.6%
lp=50%	2.2%	4.6 %	**8.9%**	7.3%
Average	4.3%	7.5 %	**10.4%**	8.7%
Synthetic data, using NB+*Gibbs*				
lp= 0%	8.5%	13.6 %	**18.6%**	15.4%
lp=10%	6.3%	9.7%	**12.7%**	10.6%
lp=50%	2.5%	3.6%	**7.4%**	5.3%
Average	5.7%	9.0%	**12.9%**	10.5%
Real data with substantial autocorrelation, using NB+ICA_C				
Cora	7.9%	**10.9%**	10.5%	-4.8%
Citeseer	15.8%	**20.5%**	14.6%	-8.3%
WebKB+co	8.3%	9.8%	**12.8%**	10.0%
HepTH	1.2%	-4.3%	3.3%	4.0%
Average	8.3%	9.2%	**10.3%**	0.2%
Other real data sets, using NB+ICA_C				
WebKB	5.8%	1.5%	-4.8%	-6.0%
Terror	2.4%	-5.7%	0.0%	0.0%

Table 9: Per-node degree results showing the amount of gain from caution (ICA_C vs. *ICA* or *Gibbs* vs. $Gibbs_{NC}$). Each value indicates the average accuracy gain from caution for all nodes in the test set within the given link degree range (nodes with degree greater than 20 were rare, and ignored for simplicity). Within each row, a value is in bold if it represents a clear peak, with monotonically decreasing accuracies to both the left and right of that value. The synthetic data used the default settings. The real data sets used the default number of attributes and lp=10%.

estimated distributions. In both cases, however, extending *ld* to even more extreme values (e.g., *ld*=0.95) does confirm the overall trend of the amount of cautious gain decreasing at high *ld*.

As expected, we found that these performance differences disappeared when many known labels were provided. In particular, at high link density and lp=50%, there were only small differences between ICA_C, ICA_{Kn}, and *ICA*, or between $wvRN_{RL}$, $wvRN_{ICA+C}$, and $wvRN_{ICA+NC}$. In addition, when PLUL was used, even *LBP* and LBP_{NC} performed on par with ICA_C and *Gibbs* when lp=50%, despite the challenges of *LBP* with high *ld*.

Overall, our results suggest that a more appropriate rendering of H3 should indicate that *the relative gain from caution will peak at some moderate value of* ld, with the precise value depending on the CC algorithm and the other data conditions. We leave confirmation of this revised hypothesis to future work.

7.6 Result 4: The Relative Gain of Cautious Inference Increases as the Labeled Proportion (*lp*) Decreases

Table 8 reports that, as *lp* varies, the regression analyses found all significant negative slopes (as expected) for the synthetic data. Likewise, in almost all cases we found significant negative slopes

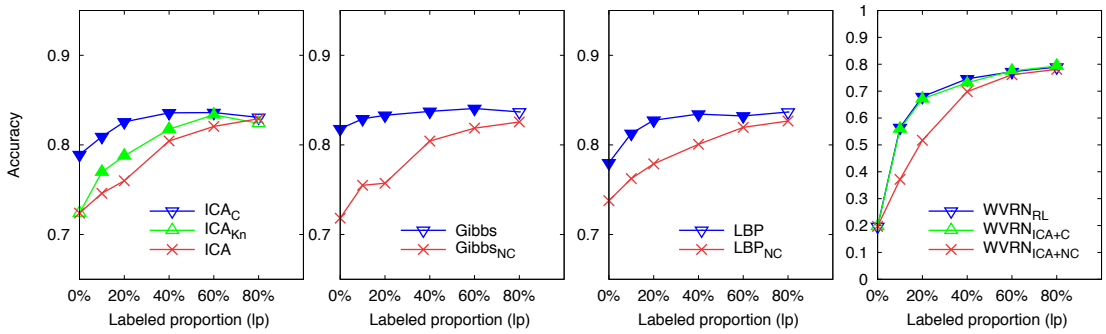


Figure 13: Results for the synthetic data as the labeled proportion (*lp*) varies.

for the real data sets with substantial autocorrelation (all except Terror and WebKB). We accept H4, because the pooled analyses find all negative, significant slopes.

For the real data, the exceptions to H4's stated trend were primarily WebKB+co, which had very erratic performance with all the algorithms, and WebKB, where none of the slopes attained statistical significance. In addition, *LBP* had highly variable behavior so that only for Citeseer did the slope approach statistical significance ($p = .053$, just over the threshold).

Figure 13, for the synthetic data, shows the performance of the cautious and non-cautious algorithms converging as *lp* increases. The cautious algorithms maintain a significant advantage until *lp*=80%. Observe that ICA_{Kn}'s curve lies between that of the more cautious ICA_C and the non-cautious *ICA*, while $wvRN_{RL}$ and $wvRN_{ICA+C}$ obtain the same results with their two different approaches to caution.

Figure 14 shows results for the real data sets as *lp* is varied. This figure show results only for *wvRN*, since results were previously presented for the other algorithms for varying numbers of attributes, and the *lp* graphs don't add additional insight for those algorithms.

The results in Figure 14 mirror those of the synthetic data, with a few exceptions. First, $wvRN_{ICA+C}$ does poorly on Terror, perhaps because of the low autocorrelation. Second, with WebKB+co, $wvRN_{ICA+C}$ outperforms $wvRN_{RL}$ when *lp* is low, though the gains are not quite significant; this effect was discussed in Section 7.5. Finally, the accuracy of *wvRN* for WebKB goes *down* with increasing *lp*. WebKB with just direct links has some autocorrelation but very low label consistency (see Table 5), because each node tends to link in certain patterns to nodes with a *different* label from itself (cf., Macskassy and Provost, 2007). Algorithms based on *wvRN* assume homophily, not such more complex forms of autocorrelation. Consequently, increasing *lp* only serves to reduce accuracy below the majority class baseline. Running *wvRN* with only co-citation links, as done for WebKB+co, works much better.

7.7 Result 5: The Relative Gain of Cautious Learning With PLUL Increases as the Labeled Proportion (*lp*) Decreases

The previous results compared cautious vs. non-cautious variants of a particular CC algorithm, in all cases using PLUL. We now justify the use of PLUL and examine its impact.

The bottom of Table 8 shows the regression slope results for H5, where the x-axis varies the labeled proportion (*lp*), and each table row compares a single CC algorithmic variant when using PLUL vs. not using PLUL. As expected, the slope analysis found all significant negative slopes for

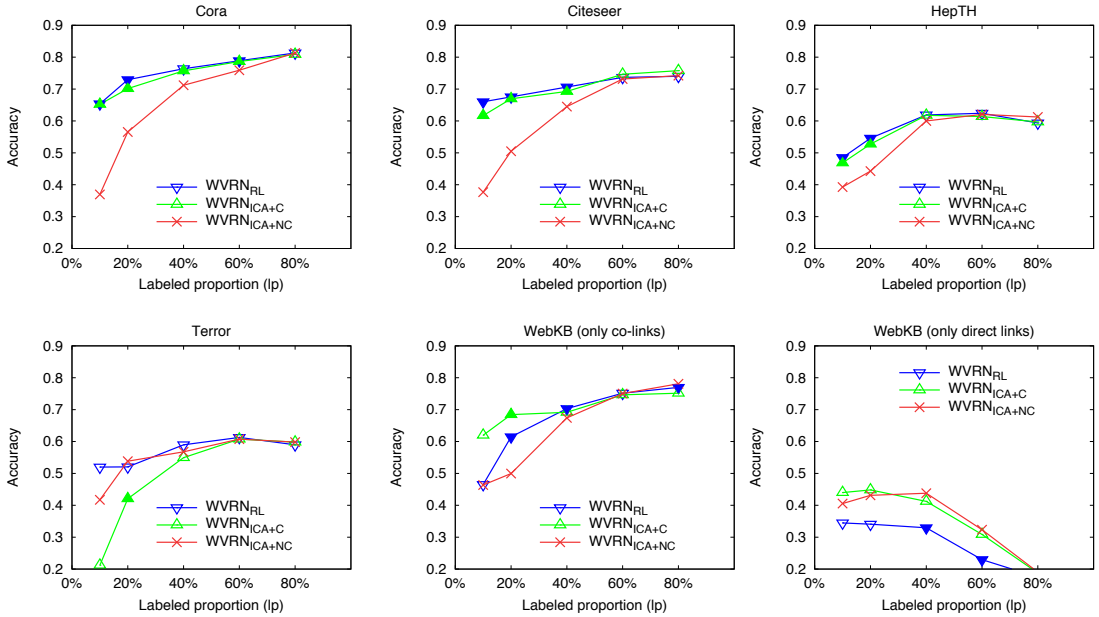


Figure 14: Results for variants of *wvRN* on the real data sets, as *lp* is varied. For the first WebKB results, *wvRN* uses *only* co-citation links (unlike previous results with other algorithms, where WebKB+co used direct links and co-links together; see Section 6.6.3). Recall that filled triangles indicate statistical significance, but only for comparing the cautious variant (here, $wvRN_{RL}$ or $wvRN_{ICA+C}$) vs. the non-cautious variant ($wvRN_{ICA+NC}$).

the synthetic data (with one exception where the p-value was close to the threshold), although the magnitude of the slopes suggests a weak trend. For the real data sets, while 18 of the 29 possible slopes were in the expected direction, only 4 of these slopes were statistically significant (recall that the real data sets have available only 3-5 trials, making significance harder to achieve). However, pooling the results across the data sets and algorithms yields a significant negative slope for both the synthetic and real data, so we accept H5.

Thus, while the effect (as *lp* varies) is smaller than with previous hypotheses, H5 indicates the PLUL provides the most gain when *lp* is small. To measure the magnitude of this gain, Table 10 shows the impact of PLUL when *lp*=0%. Each row shows the results for a different collective algorithm. Results are given for each algorithm both with and without PLUL, along with the overall gain from PLUL. Because PLUL interacts closely with the local classifier, we show results here for NB, LR, and kNN for the CC algorithms that use a local classifier. *CO* and *wvRN* are unaffected by PLUL, and thus are not shown.

In general, we found that PLUL improved performance, sometimes substantially, but the data regions where such substantial gains occur vary by classifier and/or CC algorithm. For instance, Column A of Table 10 shows results for the default synthetic data settings. Here, PLUL improves performance for almost all algorithms. In particular, the gains range from -0.3% to 10.8%, with an average of 4.0%, and are significant in 9 of the 14 cases. Column B shows results where the attribute predictiveness is 0.3 (instead of the default 0.6). In this case, the gains due to PLUL are almost

	A.) Default settings			B.) Low attr. predictiveness			C.) High link density		
	With PLUL?		PLUL	With PLUL?		PLUL	With PLUL?		PLUL
	Yes	No	Gain	Yes	No	Gain	Yes	No	Gain
Using the NB local classifier									
ICA_C	78.9	77.8	1.1	58.2	52.5	**5.7**	80.6	72.0	**8.6**
ICA	72.3	72.6	-0.3	47.7	46.8	0.9	77.0	75.4	1.6
Gibbs	81.8	81.5	0.3	60.6	55.9	**4.7**	80.8	79.2	1.6
$Gibbs_{NC}$	71.8	71.1	0.7	46.7	46.0	0.7	76.4	74.2	**2.2**
Using the LR local classifier									
ICA_C	78.6	74.1	**4.5**	56.8	43.2	**13.6**	82.9	73.9	**9.0**
ICA	70.8	68.5	**2.3**	48.5	44.4	**4.1**	70.8	72.5	-1.7
Gibbs	76.5	72.9	**3.6**	52.3	50.8	1.5	77.6	77.8	-0.2
$Gibbs_{NC}$	70.3	65.3	**5.0**	48.4	43.2	**5.2**	71.4	71.3	0.1
Using the kNN local classifier									
ICA_C	74.1	69.0	**5.1**	51.4	39.2	**12.2**	78.5	65.4	**13.1**
ICA	71.7	64.2	**7.5**	48.4	41.0	**7.4**	75.2	74.7	0.5
Gibbs	73.9	70.0	**3.9**	54.4	48.1	**6.3**	80.3	79.7	0.6
$Gibbs_{NC}$	71.7	61.3	**10.4**	47.7	38.9	**8.8**	75.0	74.0	1.0
Using *LBP*									
LBP	77.8	76.4	1.4	55.7	27.9	**27.8**	69.7	21.5	**48.2**
LBP_{NC}	73.9	63.1	**10.8**	45.5	24.4	**21.1**	54.3	31.2	**23.1**

Table 10: Impact of PLUL on accuracy with the synthetic data, for CC algorithms where PLUL applies, at lp=0%. Gains in bold are statisticaly significant.

all larger, ranging from 0.7% to 27.8% (average of 8.6%), and are significant in 11 of 14 cases. These results are consistent with H2: when attributes are less predictive of the class label, cautious techniques, including PLUL, become more important. Finally, column C shows results where the link density is now 0.7 (instead of the default 0.2); here the gains due to PLUL are more varied. For ICA_C, PLUL remains important and matters even more than with the default data settings. We conjecture that this is because with so many links, relational influence can spread very quickly in the graph, and thus the PLUL process is very important to ensuring that ICA_C's confidence measure selects the most reliable predictions during the first few iterations. Indeed, when lp is instead set to 10% (thus providing more certain estimates for the early iterations), PLUL became much less important for ICA_C. *LBP* has known issues with high link density, but PLUL helps substantially to ameliorate them. For the other algorithms, the increased link density leads to PLUL having a minor impact, consistent with H3.

Table 11 shows similar results for the real data sets, where results for all six data sets have been pooled together. Since we cannot directly vary link density, we instead show results with two conditions. On the left is the "fewer attributes" case; here each data set uses its default number of attributes, as explained in Section 6.9. On the right is the case where each data set uses 100 attributes.

Compared to results with the synthetic data, Table 11 shows less evidence for the effectiveness of PLUL with the real data sets. While all algorithms show a gain from using PLUL, only about half of the gains are statistically significant. To explain, consider that PLUL works best when the

	Fewer attributes(default)			More attributes(100)		
	With PLUL?		PLUL	With PLUL?		PLUL
	Yes	No	Gain	Yes	No	Gain
ICA_C	56.8	56.1	0.7	68.6	68.1	0.5
ICA	54.5	52.3	2.2	65.7	64.9	0.8
Gibbs	53.5	50.1	**3.4**	67.0	66.1	0.9
$Gibbs_{NC}$	55.5	53.0	2.5	66.5	65.6	**0.9**
LBP	49.9	44.3	**5.6**	65.2	58.4	**6.8**
LBP_{NC}	46.0	42.1	3.9	63.5	56.4	**7.1**

Table 11: Accuracy results showing the impact of using PLUL with the real data. Each value shows results pooled over the six real data sets, at lp=0%, using NB where applicable. Gains in bold are statistically significant.

holdout set used for learning is most similar to the test set. With the synthetic data, such similarity is likely, because the two graphs are generated from the same distribution. However, with the real data, splitting an arbitrary graph into multiple subgraphs, even while seeking to maintain similar class distributions, may nonetheless produce subgraphs with important differences (e.g., in auto-correlation), leading to sub-optimal parameter choices by PLUL. Future work is needed to explore these issues.

Nonetheless, the evidence suggests that in most cases for the real and synthetic data PLUL improves performance. Moreover, for every algorithm there was some type of data for which not using PLUL led to very poor performance. Thus, applying PLUL in all of our other experiments seemed advisable for maximizing performance and for ensuring the most equitable comparisons.

7.8 Choice of Relational Feature Types

Section 6.6 described how each trial selected a type of relational feature to use. For completeness, Table 12 summarizes how often each type of feature was chosen. In general, the best feature type (as chosen by cross-validation) varied based on the local classifier used and the data conditions. However, Table 12 shows that for NB, multiset features were dominant, especially for the more cautious algorithms (chosen 76-96% of the time for ICA_C and *Gibbs*). With kNN, proportion features were dominant, while with LR count features were chosen most often but proportion features were also fairly common, especially with high ld. These results suggest that an analyst should most likely use multiset with NB, use proportion with kNN, and consider the data conditions to select a feature type for LR.

The superiority of multiset features, when they were applicable, is interesting because they are "cautious" features that simply ignore nodes with no known or predicted label (see Section 6.6.1). Likewise, Section 6.5 reported that LR with ICA_C performed best when missing feature values were ignored (by using a separate classifier trained without the missing features). These results are consistent with Saar-Tsechansky and Provost (2007), who found (for non-relational data) this "reduced-feature model" approach to be superior to commonly used approaches based on imputation. For a non-relational setting, their results thus demonstrate the superiority of a more "cautious" approach to handling missing values during testing. For relational domains, we could imagine taking this idea of ignoring missing/estimated values even further, e.g., using a classifier that ignored

	A.) ICA_C			B.) *ICA*			C.) *Gibbs*		
	Mult.	Count	Prop.	Mult.	Count	Prop.	Mult.	Count	Prop.
Synthetic data, using the NB local classifier									
Default	96%	0%	4%	72%	0%	28%	100%	0%	0%
Low *ap*	88%	0%	12%	20%	4%	76%	92%	0%	8%
High *ld*	48%	0%	52%	80%	0%	20%	96%	0%	4%
Average	77%	0%	23%	57%	1%	41%	96%	0%	4%
Synthetic data, using the LR local classifier									
Default	n.a.	92%	8%	n.a.	80%	20%	n.a.	80%	20%
Low *ap*	n.a.	52%	48%	n.a.	60%	40%	n.a.	68%	32%
High *ld*	n.a.	80%	20%	n.a.	52%	48%	n.a.	48%	52%
Average	n.a.	75%	25%	n.a.	64%	36%	n.a.	65%	35%
Synthetic data, using the kNN local classifier									
Default	n.a.	0%	100%	n.a.	0%	100%	n.a.	0%	100%
Low *ap*	n.a.	0%	100%	n.a.	12%	88%	n.a.	0%	100%
High *ld*	n.a.	0%	100%	n.a.	0%	100%	n.a.	0%	100%
Average	n.a.	0%	100%	n.a.	4%	96%	n.a.	0%	100%
Real data, using the NB local classifier									
Cora	97.5%	2.5%	0.0%	70.0%	17.5%	12.5%	100.0%	0.0%	0.0%
Citeseer	92.5%	2.5%	5.0%	57.5%	32.5%	10.0%	100.0%	0.0%	0.0%
WebKB+co	84.4%	0.0%	15.6%	65.6%	34.4%	0.0%	71.9%	12.5%	15.6%
WebKB	53.1%	40.6%	6.3%	31.3%	56.3%	12.5%	75.0%	21.9%	3.1%
HepTH	85.0%	12.5%	2.5%	62.5%	25.0%	12.5%	70.0%	27.5%	2.5%
Terror	50.0%	8.3%	41.7%	50.0%	25.0%	25.0%	41.7%	16.7%	41.7%
Average	77.1%	11.1%	11.8%	56.1%	31.8%	12.1%	76.4%	13.1%	10.5%

Table 12: The relational feature type (multiset, count, or proportion) chosen by cross-validation. For the synthetic data, results are shown with the default settings, with low attribute predictiveness (*ap*=0.3), and with high link density (*ld*=0.7). For the real data, results are shown averaged across all the data points shown in Figures 10 and 11.

the estimated label of a linked node but instead directly used its non-relational features. However, Jensen et al. (2004) demonstrated that such an approach is generally inferior to the approaches we consider in this article (label-based features with collective inference), because of the much larger number of model parameters that must be learned for the former case.

7.9 Variants of *wvRN*

Most prior research involving *wvRN* has used $wvRN_{RL}$, the variant suggested as a relational-only baseline by Macskassy and Provost (2007). However, algorithms based on *wvRN* need not necessarily be relational-only. For instance, Macskassy (2007) described a technique for adding additional links to the graph between nodes that appeared similar based on their attributes. Alternatively, we could imagine, for $wvRN_{RL}$, initializing each node's predicted label probabilities based upon the output of an attribute-only local classifier (instead of using class priors as done in Figure 5). Unfortunately, this idea does not work well for a "soft" algorithm such as $wvRN_{RL}$, because after iterating many times the current state is almost completely determined by the known labels, independent

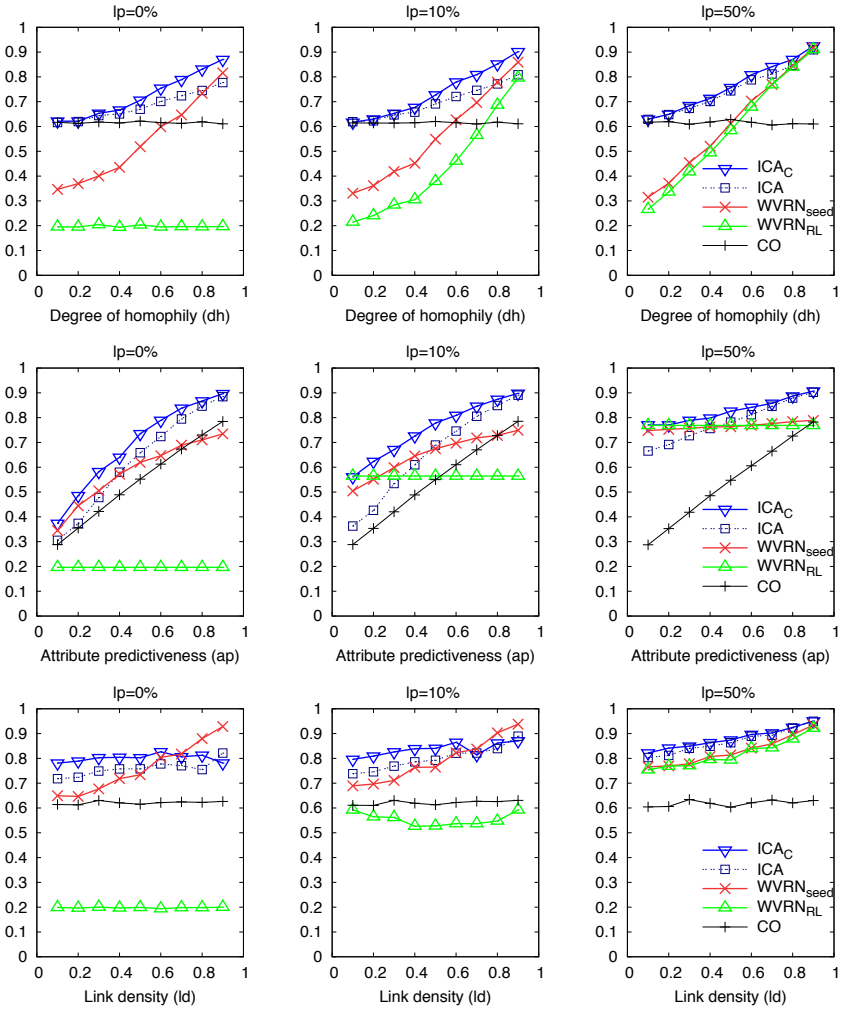


Figure 15: Results for the synthetic data where $wvRN_{seed}$ is added for comparison. Because of the multiple possible comparisons, filled triangles are not used here to indicate statistical significance.

of the starting state (Macskassy and Provost, 2005). While in principle this problem could be addressed via learning an appropriate decay parameter Γ and stopping point, this forfeits much of the simplicity of *wvRN*.

In contrast to $wvRN_{RL}$, with a hard-labeling algorithm such as $wvRN_{ICA+C}$, the initial conditions do matter. In particular, we evaluated $wvRN_{seed}$, an algorithm that behaves just like $wvRN_{ICA+C}$, except that each node's predicted label is initialized to the most likely label predicted by an attribute-only NB classifier. Non-relational information thus "seeds" the inference process but is then not explicitly used again. To the best of our knowledge, this algorithm has not been previously considered for CC.

Figure 15 shows a variety of results for the synthetic data; results with the real data showed similar trends. Overall, $wvRN_{seed}$ outperforms $wvRN_{RL}$ (especially when *lp* is low), which is to be expected since $wvRN_{seed}$ uses more information. $wvRN_{seed}$ generally underperforms ICA_C, which

is also to be expected since ICA_C both uses predicted labels cautiously (while $wvRN_{seed}$ treats all predictions equally) and continues to use both attribute and relational information after the first iteration. The differences with ICA_C are largest when *dh* is low (where *wvRN*'s homophily assumption is violated) or when attribute predictiveness is high (since $wvRN_{seed}$ uses the attributes only at initialization). However, $wvRN_{seed}$ outperforms all of the other shown algorithms when link density is high. This case is analogous to the results with ICA_{Kn} from Section 7.5: if accuracy and link density are high (and homophily is present), then caution with relational information may not be necessary, and this case shows that continuing to use non-relational information after initialization may also not be necessary. Overall, the results indicate that $wvRN_{seed}$ is not likely to be a strong contender as a general purpose CC algorithm, but they do demonstrate an effective way to add non-relational information to *wvRN*-based algorithms.

7.10 Impact of the Default Values for Synthetic Data Generation

The synthetic data evaluated above was generated with the default parameters described in Table 4. Conceivably, our choice of default values could have an important effect on the results. While our evaluation of multiple real data sets has already helped to validate the synthetic data results, we also carried out an extensive exploration with other default values. For instance, when varying *ld*, we experimented with all combinations of *ap*= $\{0.4, 0.6, 0.8\}$, *dh*= $\{0.5, 0.7, 0.9\}$, and *lp*=$\{0\%, 10\%, 50\%\}$. For tractability, we only evaluated variants of *ICA*, since the above results show that ICA_C produced the best or nearly the best results for all synthetic and real data sets, and that other cautious algorithms usually behaved like ICA_C.

The trends were highly consistent with the results we report and agree with our accepted hypotheses. For instance, if the default *ap* is very high, the results for varying *dh* showed a much smaller slope for the relative impact of cautious ICA_C vs *ICA*. The only default value that noticeably changed any result was already reported in Section 7.5: when *ap* was small (e.g., 0.4), the unusual advantage of *ICA* over ICA_C observed at very high *ld* disappeared. Thus, we believe the trends in our results are robust over a wide range of data characteristics.

8. Discussion

In this section we compare results with different families of algorithms, examine the overall effectiveness of caution, and use our results to explain the findings of some previous research.

8.1 Comparisons Across Algorithmic Families

Section 7 focused on comparing cautious vs. non-cautious variants within the same algorithmic family. We now briefly compare across these families. We focus on the algorithms that have been most frequently used in previous work: *ICA*, *Gibbs*, *LBP*, and $wvRN_{RL}$. We also include the less studied ICA_C, since our results show that it has very strong performance. We report specific results for *lp*=10%; comparisons were similar for *lp*=0%, while all of the algorithms perform very similarly when *lp*=50%.

$wvRN_{RL}$'s performance depends on homophily, link density, and *lp*. In our study, $wvRN_{RL}$ was thus competitive with the other CC algorithms when homophily and/or *lp* was high, or when the attributes were not very predictive. On the other hand, $wvRN_{RL}$ requires that some labels are known

in the test set, so it is not applicable when *lp*=0% (the out-of-sample task). $wvRN_{seed}$ would be an alternative.

For the synthetic data, the cautious algorithms ICA_C, *Gibbs*, and *LBP* had remarkably similar performance. Among the three, *Gibbs* had a small but sometimes significant performance advantage. For instance, across the results for varying *dh* at *lp*=10% shown in Figure 8, *Gibbs* outperformed ICA_C by an average of 1.0% (significantly for $dh \geq 0.6$) and *LBP* by an average of 0.7% (significantly for $0.4 \leq dh \leq 0.7$). Neither ICA_C nor *LBP* had consistent, significant gains over the other, except that both *Gibbs* and ICA_C had substantial, significant gains over *LBP* when attribute strength was very low (gains of 5-8%) or when link density was high (gains of 14-25%). However, all three algorithms did have substantial, significant gains vs. *ICA*, except for when *dh* was very low or when *ld* was very high. For instance, across the various *dh* levels, *Gibbs* outperformed *ICA* by 0.9-11.2% (all significantly) except for a loss of 0.1% at *dh*=0.1. Thus, based on the synthetic data results, ICA_C, *Gibbs*, and *LBP* usually achieve similar accuracies, despite their use of very different approaches to caution.

On the real data sets, ICA_C, *Gibbs*, and *LBP* likewise performed similarly. However, there are two kinds of differences that should be noted. First, there were a few data sets on which *LBP* and/or *Gibbs* performed noticeably worse than ICA_C. In particular, *Gibbs* has poor performance on HepTH and WebKB+co. In both cases, this is likely due to issues of high link density (WebKB has very many co-citation links; HepTH has fewer links but some nodes have very high degree). High link density can lead to extreme probabilities, where *Gibbs* is known to perform poorly. While this was not a particular problem with the synthetic data (perhaps because the training and test graphs were more similar), NB is well known for producing polarized probabilities in some cases. PLUL does help, for instance, improving performance on HepTH and WebKB+co by an average of 4% and 15%, respectively, in Figures 10 and 11. Nonetheless, performance with *Gibbs* lags that of ICA_C or *ICA*, which are not so influenced by extreme probabilities. We experimented with more and/or longer *Gibbs* chains but this did not improve performance. However, this is one case where the LR classifier performed better than NB: it appears to produce less polarized probabilities than NB, leading to improved performance with *Gibbs* (see Figures 24 and 27 in the online appendix). Similarly, *LBP*, which struggles with high link density, also has problems with HepTH (and likely would have low performance with WebKB+co, had it ever converged) and with Cora. Its difficulty with Cora is surprising and possibly indicates that the conjugate gradient training did not perform adequately, despite our attempts (cf., Sen et al., 2008). However, *LBP* did perform well on Citeseer, which has similar characteristics.

Second, in contrast to the small advantage for *Gibbs* on the synthetic data, for the real data ICA_C holds a small advantage. For instance, in Figure 10, ICA_C outperforms *Gibbs* on average by 1% for Cora and 2.4% for Citeseer, though not significantly. For HepTH and WebKB+co, where *Gibbs* had trouble, the gains averaged 5.4% and 21.0%, respectively, and were significant for HepTH when the number of attributes was small. ICA_C was also robust: it was the only algorithm to outperform *ICA* on average for every real data set considered. Moreover, using results pooled over all six data sets, ICA_C had moderate gains vs. *ICA*, *Gibbs*, *LBP*, and $wvRN_{RL}$, both at the default number of attributes (where the gains were significant) and using 100 attributes for each data set. Comparing to just *Gibbs* and *LBP*, ICA_C had a pooled gain of 4.9% and 7.8%, respectively, with the default number of attributes, and 1.8% and 4.5%, respectively, with 100 attributes.

8.2 Cautious Behavior as a Predictor of Performance

The previous section identified some of the situations in which the algorithms performed similarly or differently. However, if we exclude the extreme data conditions such as very low attribute predictiveness or high link density, a more remarkable finding emerges: *the amount of cautious inference used by an algorithm strongly predicts its relative performance*. This finding is especially interesting because the precise type of cautious inference seems to matter little. On both the synthetic and the real data sets, in most cases ICA_C, *Gibbs*, and *LBP* perform alike, while the non-cautious *ICA*, $Gibbs_{NC}$, and LBP_{NC} also perform similarly to each other (and at lower accuracy levels than the cautious algorithms). However, when many test labels are known (high *lp*), the need for caution decreases, and the differences between these two groups greatly diminish.

This effect can also be seen in other CC variants. For instance, $wvRN_{RL}$ and $wvRN_{ICA+C}$ perform similarly, despite their very different approaches to caution, and they both outperform the non-cautious $wvRN_{ICA+NC}$. Likewise, in almost every case the somewhat-cautious ICA_{Kn} attained an accuracy between that of the more cautions ICA_C and the non-cautious *ICA*.

Thus, the amount of cautious inference seems to be the biggest factor differentiating those algorithms that use attributes, much more so than whether some kind of *ICA* or *Gibbs* or *LBP* is used. Likewise, when attributes are not used, as with the variants of *wvRN*, caution also appears to be the largest factor in predicting relative performance.

8.3 Limitations of Cautious Inference

While our results show that the cautious use of relational information can significantly boost performance, adding more caution to an algorithm is not always beneficial. In particular, the most extreme form of relational caution is to not use any relational information (i.e., *CO*), but that is seldom optimal. Instead, an algorithm must seek to cautiously avoid errors from noisy predictions while still leveraging informative relations.

To illustrate these effects, Figure 16 shows accuracy results for three synthetic data conditions: low attribute predictiveness (*ap*=0.3), the default settings, and high link density (*ld*=0.9). Here the x-axis indicates the algorithm used, with the amount of relational caution used increasing to the right. We focus on variants of *ICA*, but add three new algorithms for further analysis. ICA_{70} is just like ICA_C, except that it stops after it has "committed" and used the most certain 70% of the predicted labels (i.e., after the iteration when $h = 7$ in Figure 2). ICA_{30} and ICA_0 likewise stop after accepting and using 30% and 0% of the predicted labels, respectively. Note that ICA_0 is identical to ICA_{Kn} during the very first iteration (when both use only the "known" labels for relational features), but that ICA_0 stops after that iteration, while ICA_{Kn} continues for 10 more iterations, using all available predictions during those iterations.

For the default and low attribute predictiveness data conditions, the trends are very similar: amongst *ICA*, ICA_{Kn}, and ICA_C, the most cautious ICA_C performs best. Adding more caution to ICA_C, however, consistently decreases performance, as ICA_{70}, ICA_{30}, and ICA_0 use less and less relational information, until the lowest performance is found with the non-relational *CO*. These results make sense: for this data, relational links *are* informative, so completely ignoring any (or all) of them is non-optimal. Indeed, using all of them without any caution (*ICA*) is much better than cautiously ignoring all relations (*CO*), but the cautious algorithm that eventually uses all relations (ICA_C) performs best. Note that this property of (eventually) using all available relational informa-

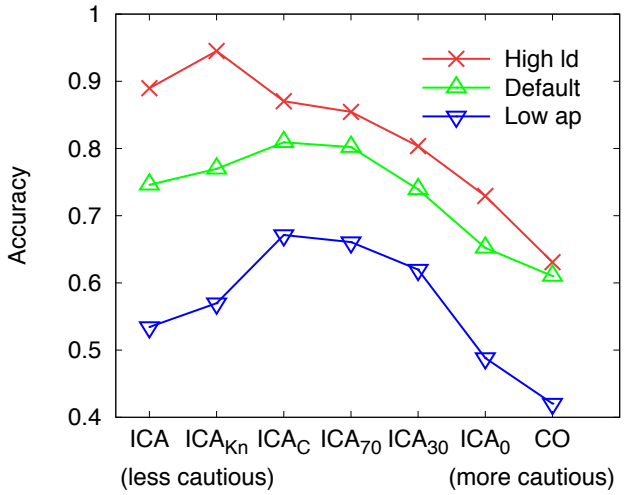


Figure 16: Accuracy as a function of the amount of relational caution used. ICA_{70}, ICA_{30}, and ICA_0 are (even more cautious) variants of ICA_C that stop iterating before some of the less certain relational information has been used.

tion is true of all of the more cautious algorithms that we considered in this article (ICA_C, *Gibbs*, *LBP*, $wvRN_{RL}$, and $wvRN_{ICA+C}$).

The high link density case provides an interesting contrast. Here the general shape of the curve is similar, but the peak performance is observed with ICA_{Kn}, not with the more cautious ICA_C. This effect was already discussed in Section 7.5: if the baseline accuracy is high and there are many links, simply using all available information after the first iteration is best. Similarly, for situations where caution is not very important (e.g., when *lp* is high), the curve would show similar results for *ICA*, ICA_{Kn}, and ICA_C. Thus, in most cases being cautious with relational information is best, but the algorithm should eventually use all available information (relational and non-relational), and in some cases using more caution may be less important or even harmful.

8.4 Explanation of Prior Results

Our investigation enables us to explain the questions from Section 1, among others:

1. **Why did Sen et al. (2008) find no consistent difference between *Gibbs* and *ICA*?** In contrast, *Gibbs* had worked well in other work, and in this article we found that *Gibbs* (and ICA_C) often significantly increases accuracy vs. *ICA*. However, our results and careful study of Sen et al.'s methodology explains the discrepancy: to generate the test set, they used a snowball sampling method that we found produces an effective labeled proportion (*lp*) of at least 0.5—a region where the use of caution has little impact. Also, their study did not vary attribute predictiveness, which we show is a significant factor in the relative performance of more cautious CC algorithms.

2. **Why did McDowell et al. (2007a) find that ICA_C significantly outperforms *Gibbs*, even though attribute predictiveness was high, while here we find that *Gibbs* performs on**

par or better than ICA_C in such cases? To investigate, we re-ran our experiments from our earlier paper, but with two variations informed by our now-refined understanding of CC. First, we used PLUL with both the NB and kNN classifiers. Second, we changed the NB classifier to use multiset relational features (instead of proportion), which use more information and which Section 7.8 shows is the feature of choice when using NB (it didn't apply for kNN). With these enhancements, *Gibbs*'s relative performance improved, so that ICA_C and *Gibbs* both significantly outperformed *ICA*, but the results for *Gibbs* and ICA_C did not significantly differ. Thus, more careful learning and representation choices resolves the discrepancy. This also suggests that not using PLUL could potentially have an important effect on performance comparisons. As an additional example, Sen and Getoor (2006) experimented with a wide range of link densities but did not use a technique like PLUL; our results suggest that using PLUL could have significantly improved their results with *LBP* for high *ld*.

3. **Why did Galstyan and Cohen (2007) find that a soft-labeling version of *wvRN* fails to consistently outperform a hard "label propagation" (LP) version?** Most authors have expected that, for relational-only classification, the soft-labeling algorithm that directly reasons with probabilities (thus exercising cautious inference) should outperform a hard-labeling version that only reasons with the single most likely label for each linked node. However, closer examination of their LP algorithm reveals that it includes elements of caution. In particular, after each iteration, LP labels a non-known node only if the estimated score for that node is among the *highest* of any such nodes. Thus, in a way similar to $wvRN_{ICA+C}$, nodes that are closest to known nodes are labeled first, and the algorithm effectively favors label information that was either known or is closer to other known nodes. This cautious behavior enables LP to be competitive with (and sometimes outperform) the soft-labeling algorithm.

4. **Why did Sen et al. (2008) find that *ICA* and *Gibbs* perform better with LR than with NB, while we find the reverse?** We replicated the synthetic data of their paper, and reproduced their results. A key point, however, is that Sen et al. used count relational features for both NB and LR, while we used cross-validation on a holdout set to select the best relational feature type (see Section 6.6). This procedure predominantly selected multiset features for NB (see Section 7.8), which we found in separate experiments to consistently improve NB performance compared to using count features. Consequently, in our results CC algorithms that use NB almost always outperformed those that use LR. While not a focus of our work, such differences can be seen in Table 10. The superior performance of multiset features also confirms the finding of Neville et al. (2003b).

5. **When will cautious algorithms outperform their aggressive variants?** We found that using more cautious CC frequently and sometimes dramatically increased accuracy. In general, cautious CC performs comparatively well whenever relational inference errors are more likely. These errors occur more frequently when there is more uncertainty in the estimated relational feature values (e.g., when the attribute predictiveness is low) or when the effect of any such uncertainty is magnified (e.g., when autocorrelation is high). In some cases, such as when the test set links to many known labels (high *lp*), using a more cautious CC algorithm may be unnecessary. However, in many cases (and with most previous work) *lp* is small or zero, and thus caution may be important.

9. Conclusion

Collective classification's greatest strength—making inferences based on the inferred labels of related nodes—can also be a significant weakness, since this use of uncertain labels may reduce accuracy when the estimates are incorrect. In this article, we demonstrated that managing this estimation uncertainty through "cautious" algorithmic behavior is essential to achieving maximal, robust performance. We showed how varying degrees of cautious inference could be manifested in four different collective inference families, and explained how to use cautious learning with PLUL to further improve performance. Our experimental results with both synthetic and real-world data sets showed that cautious algorithms did outperform their non-cautious variants. By exploring a wide range of data, we identified some data characteristics for which this performance advantage grew larger. In particular, cautious behavior is especially important when there is a higher probability of incorrect relational inference—which occurs when autocorrelation is higher, when link density is moderate, and/or when attribute predictiveness or the labeled proportion is lower. In addition, our study enabled us to answer several important questions from previous work.

Across a wide range of data, we found that an algorithm's degree of caution was a significant predictor of relative performance—in most cases a more important one than the specific collective inference algorithm used. This reinforces the fundamental importance of cautious behavior for CC. However, the cautious CC algorithms were not always comparable. *Gibbs* and (especially) *LBP* sometimes struggled (e.g., when the data had high link density). In contrast, ICA_C was a very reliable performer and almost always had maximal or near-maximal performance, especially for the real-world data. This finding is interesting because this article is the first to consider ICA_C in depth. Moreover, ICA_C is a simple modification to *ICA*, making it much more time-efficient than *Gibbs* or *LBP*. This suggests that ICA_C is a strong contender for general CC tasks, and should be used as a baseline for future CC performance comparisons.

Regarding cautious learning, we found that PLUL generally increased accuracy, sometimes substantially. Parameter tuning is known to be important for learning non-relational classifiers. We show that it can be especially critical for CC due to CC's reliance on uncertain labels during testing. For example, further results showed that for the synthetic data when link density was high, *Gibbs*+NB with a naive α (prior hyperparameter) of 1.0 attained 99% of the accuracy attainable with any α—*if* most test labels were known (e.g., *lp*=80%). However, when *lp*=0% this strategy's accuracy was just 61% of optimal. Using PLUL to set α instead increased accuracy. In addition, our results in Section 7.7 showed PLUL helping both cautious and non-cautious inference algorithms. Thus, using PLUL for cautious learning improves performance, and adding cautious inference helps even more.

Future work is needed to compare the algorithms considered here with alternative methods, such as Markov Logic Networks (Richardson and Domingos, 2006) and the "ghost edge" approach of Gallagher et al. (2008), and to compare PLUL to the alternative "stacked models" discussed in Section 5.5. In addition, further studies to consider the effect of training set size, noise in the known labels, and link uncertainty would be useful. Finally, techniques are needed to further improve the performance of cautious inference on data with high link density or other extreme conditions.

Acknowledgments

Thanks to Doug Downey, Lise Getoor, David Jensen, and Sofus Macskassy for helpful comments on this work, to Prithviraj Sen for the Cora and Citeseer data sets, to Jennifer Neville for helpful discussions and for code that implements LBP, and to Prithviraj Sen and Mustafa Bilgic for clarifications on their work. Thanks also to the anonymous reviewers for many helpful comments that helped to improve this article. Luke McDowell's funding for this research was partly supported by the U.S. Naval Academy Cooperative Program for Scientific Interchange, which is a component of NRL's General Laboratory Scientific Interchange Program. Portions of this analysis were conducted using Proximity, an open-source software environment developed by the Knowledge Discovery Laboratory at the University of Massachusetts Amherst (http://kdl.cs.umass.edu/proximity/). The HepTH data was derived from the Proximity HEP-Th database, which is based on data from the arXiv archive and the Stanford Linear Accelerator Center SPIRES-HEP database provided for the 2003 KDD Cup competition, with additional preparation performed by the Knowledge Discovery Lab.

Appendix A. Measuring the Strength of Relational Dependence

Data sets used for CC are often measured for their autocorrelation. Alternatively, *label consistency* is the percentage of links connecting nodes with the same label. A closely related measure is the *degree of homophily* (*dh*) used by Sen et al. (2008). To see the difference, suppose that a data set has five labels that occur with equal frequency. Sen et al. argue that, if *dh* is zero, the target of a link from a node labeled *A* should be to another node labeled *A* 20% of the time (random chance), not 0% of the time (Sen, 2008). Thus, for a uniform class distribution, the actual probability of a link connecting two nodes i and j of the same label is defined as:

$$\text{label consistency} = P(y_i = y_j | (i,j) \in E) = dh + \frac{1-dh}{|C|}. \tag{4}$$

To facilitate comparison, we adopt this definition to generate synthetic data with varying levels of *dh*. However, for real data sets, we can only directly compute label consistency. Thus, to facilitate comparison we also compute *approximate homophily* from the measured label consistency by assuming a uniform distribution of labels and solving for *dh* using Equation 4.

Appendix B. Information on Additional Results

In Section 7, we omitted some results for alternate local classifiers (LR and kNN) and/or alternate settings of *lp*, since they did not noticeably change our reported trends. These results are available in an online appendix that accompanies this article on the JMLR website.

References

Regina Barzilay and Mirella Lapata. Collective content selection for concept-to-text generation. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 331–338, 2005.

Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48(3):259–302, 1986.

Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society*, 36(2):192–236, 1974.

Mustafa Bilgic and Lise Getoor. Effective label acquisition for collective classification. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 43–51, 2008.

Béla Bollobás, Christian Borgs, Jennifer Chayes, and Oliver Riordan. Directed scale-free graphs. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 132–139, 2003.

Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the 15th Conference of the American Association for Artificial Intelligence (AAAI)*, pages 509–516, 1998.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

Roland L. Dobrushin. The description of a random field by means of conditional probabilities and conditions of its regularity. *Theory of Probability and its Applications*, 13(2):197–224, 1968.

Andrew Fast and David Jensen. Why stacked models perform effective collective classification. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2008.

Karen Yuen Fung and Barbara A. Wrobel. The treatment of missing values in logistic regression. *Biometrical Journal*, 31(1):35–47, 1989.

Brian Gallagher and Tina Eliassi-Rad. Leveraging label-independent features for classification in sparsely labeled networks: An empirical study. In *Proceedings of the 2nd Workshop on Social Network Mining and Analysis at the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

Brian Gallagher, Hanghang Tong, Tina Eliassi-Rad, and Christos Faloutsos. Using ghost edges for classification in sparsely labeled networks. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 256–264, 2008.

Aram Galstyan and Paul R. Cohen. Empirical comparison of "hard" and "soft" label propagation for relational classification. In *Proceedings of the 17th International Conference on Inductive Logic Programming (ILP)*, pages 98–111, 2007.

Tayfun Gurel and Kristian Kersting. On the trade-off betweeen iterative classification and collective classification: first experimental results. In *Working Notes of the 3rd International ECML/PKDD Workshop on Mining Graphs, Trees, and Sequences*, 2005.

David Heckerman. A tutorial on learning with bayesian networks. In M. Jordan, editor, *Learning in Graphical Models*. MIT Press, 1999.

Andreas Heß and Nicholas Kushmerick. Iterative ensemble classification for relational data: A case study of semantic web services. In *Proceedings of the 15th European Conference on Machine Learning (ECML)*, pages 156–167, 2004.

Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

David Jensen and Jennifer Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 259–266, 2002.

David Jensen, Jennifer Neville, and Michael Hay. Avoiding bias when aggregating relational data with degree disparity. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 274–281, 2003.

David Jensen, Jennifer Neville, and Brian Gallagher. Why collective inference improves relational classification. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 593–598, 2004.

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

Daphne Koller, Nir Friedman, Lise Getoor, and Benjamin Taskar. Graphical models in a nutshell. In L. Getoor and B. Taskar, editors, *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

Zhenzhen Kou and William W. Cohen. Stacked graphical models for efficient inference in Markov Random Fields. In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM)*, pages 533–538, 2007.

Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 496–503, 2003a.

Qing Lu and Lise Getoor. Link-based classification using labeled and unlabeled data. In *Proceedings of the Workshop on the Continuum from Labeled to Unlabeled data at the 20th International Conference on Machine Learning (ICML)*, 2003b.

Sofus A. Macskassy. Improving learning in networked data by combining explicit and mined links. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 590–595, 2007.

Sofus A. Macskassy and Foster Provost. Suspicion scoring based on guilt-by-association, collective inference, and focused data access. In *Proceedings of the International Conference on Intelligence Analysis*, 2005.

Sofus A. Macskassy and Foster Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983, 2007.

Sofus A. Macskassy and Foster Provost. A brief survey of machine learning methods for classification in networked data and an application to suspicion scoring. In *Proceedings of the Workshop on Statistical Network Analysis at the 23rd International Conference on Machine Learning (ICML)*, 2006.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 591–598, 2000a.

Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000b.

Luke K. McDowell, Kalyan Moy Gupta, and David W. Aha. Cautious inference in collective classification. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 596–601, 2007a.

Luke K. McDowell, Kalyan Moy Gupta, and David W. Aha. Case-based collective classification. In *Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 399–404, 2007b.

Robert J. McEliece, David J. C. MacKay, and Jung-Fu Cheng. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16 (2):140–152, 1998.

Miller McPherson, Lynn Smith-Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27:415–444, 2001.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

Jennifer Neville and David Jensen. A bias/variance decomposition for models using collective inference. *Machine Learning Journal*, 73(1):87–106, 2008.

Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proceedings of the Workshop on Learning Statistical Models from Relational Data at the 17th National Conference on Artificial Intelligence (AAAI)*, pages 13–20, 2000.

Jennifer Neville and David Jensen. Leveraging relational autocorrelation with latent group models. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*, pages 170–177, 2005.

Jennifer Neville and David Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007.

Jennifer Neville, David Jensen, Lisa Friedland, and Michael Hay. Learning relational probability trees. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 625–630, 2003a.

Jennifer Neville, David Jensen, and Brian Gallagher. Simple estimators for relational bayesian classifiers. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM)*, pages 609–612, 2003b.

Jennifer Neville, Özgür Simsek, David Jensen, John Komoroske, Kelly Palmer, and Henry G. Goldberg. Using relational knowledge discovery to prevent securities fraud. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 449–458, 2005.

Mark E. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, 2003.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

Matthew J. Rattigan, Marc Maier, David Jensen, Bin Wu, Xin Pei, JianBin Tan, and Yi Wang:. Exploiting network structure for active inference in collective classification. In *Proceedings of the Workshop on Mining Graphs and Complex Structures at the 7th IEEE International Conference on Data Mining (ICDM)*, pages 429–434, 2007.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2): 107–136, 2006.

Maytal Saar-Tsechansky and Foster Provost. Handling missing values when applying classification models. *Journal of Machine Learning Research*, 8(Jul):1623–1657, 2007.

Prithviraj Sen. Personal communication, 2008.

Prithviraj Sen and Lise Getoor. Empirical comparison of approximate inference algorithms for networked data. In *Proceedings of the Workshop on Open Problems in Statistical Relational Learning at the 23rd International Conference on Machine Learning (ICML)*, 2006.

Prithviraj Sen and Lise Getoor. Link-based classification. Technical Report CS-TR-4858, University of Maryland, College Park, MD, February 2007.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine, Special Issue on AI and Networks*, 29(3): 93–106, 2008.

Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probalistic models for relational data. In *Proceedings of the 18th Conference on Uncertainity in Artificial Intelligence (UAI)*, pages 485–492, 2002.

YongHong Tian, Tiejun Huang, and Wen Gao. Latent linkage semantic kernels for collective classification of link data. *Journal of Intelligent Information Systems*, 26(3):269–301, 2006.

Rudolph Triebel, Richard Schmidt, Oscar Martinez Mozos, and Wolfram Burgard. Instance-based AMN classification for improved object recognition in 2D and 3D laser range data. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2225–2230, 2007.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Generalized belief propagation. *Advances in Neural Information Processing Systems (NIPS)*, 13:689–695, 2000.

Nevin Lianwen Zhang and David Poole. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.

Bin Zhao, Prithviraj Sen, and Lise Getoor. Event classification and relationship labeling in affiliation networks. In *Proceedings of the Workshop on Statistical Network Analysis (SNA) at the 23rd International Conference on Machine Learning (ICML)*, 2006.

Journal of Machine Learning Research 10 (2009) 2837-2871 Submitted 4/08; Revised 2/09; Published 12/09

Adaptive False Discovery Rate Control under Independence and Dependence

Gilles Blanchard GILLES.BLANCHARD@WIAS-BERLIN.DE
Weierstraß Institut für angewandte Analysis und Stochastik
Mohrenstrasse 39
10117 Berlin, Germany

Étienne Roquain ETIENNE.ROQUAIN@UPMC.FR
UPMC Univ Paris 06, UMR 7599, Laboratoire de Probabilités et Modèles Aléatoires (LPMA)
4 Place Jussieu,
75252 Paris cedex 05, France

Editor: John Shawe-Taylor

Abstract

In the context of multiple hypothesis testing, the proportion π_0 of true null hypotheses in the pool of hypotheses to test often plays a crucial role, although it is generally unknown *a priori*. A testing procedure using an implicit or explicit estimate of this quantity in order to improve its efficency is called *adaptive*. In this paper, we focus on the issue of false discovery rate (FDR) control and we present new adaptive multiple testing procedures with control of the FDR. In a first part, assuming independence of the *p*-values, we present two new procedures and give a unified review of other existing adaptive procedures that have provably controlled FDR. We report extensive simulation results comparing these procedures and testing their robustness when the independence assumption is violated. The new proposed procedures appear competitive with existing ones. The overall best, though, is reported to be Storey's estimator, albeit for a specific parameter setting that does not appear to have been considered before. In a second part, we propose adaptive versions of step-up procedures that have provably controlled FDR under positive dependence and unspecified dependence of the *p*-values, respectively. In the latter case, while simulations only show an improvement over non-adaptive procedures in limited situations, these are to our knowledge among the first theoretically founded adaptive multiple testing procedures that control the FDR when the *p*-values are not independent.

Keywords: multiple testing, false discovery rate, adaptive procedure, positive regression dependence, *p*-values

1. Introduction

The topic of multiple testing, which enjoys a long history in the statistics literature, has generated a renewed, growing attention in the recent years, spurred by an increasing number of application fields, in particular bioinformatics. For example, when processing microarray data, a common goal is to detect which genes (among several ten of thousands) exhibit a significantly different level of expression in two different experimental conditions. Each gene represents a "hypothesis" to be tested in the statistical sense. The genes' expression levels fluctuate naturally (not to speak of other sources of fluctuation introduced by the experimental protocol), and, because the number of candidate genes is large, it is important to control precisely what can be deemed a significant

observed difference. Generally, it is assumed that the natural fluctuation distribution of a *single* gene is known and the problem is to take into account the number of genes involved (for more details, see for instance Dudoit et al., 2003).

1.1 Adaptive Multiple Testing Procedures

In this work, we focus on building multiple testing procedures with a control of the false discovery rate (FDR). This quantity is defined as the expected proportion of type I errors, that is, the proportion of true null hypotheses among all the null hypotheses that have been rejected (i.e., declared as false) by the procedure. In their seminal work on this topic, Benjamini and Hochberg (1995) proposed the celebrated *linear step-up* (LSU) procedure, which was proved to control the FDR under the assumption of independence between the p-values. Later, it was proved (Benjamini and Yekutieli, 2001) that the LSU procedure still controls the FDR when the p-values have positive dependence (or more precisely, a specific form of positive dependence called positive regression dependence from a subset, PRDS). Under completely unspecified dependence, the same authors have shown that the FDR control still holds if the threshold collection of the LSU procedure is divided by a factor $1+1/2+\cdots+1/m$, where m is the total number of null hypotheses to test. More recently, the latter result has been generalized (Blanchard and Fleuret, 2007; Blanchard and Roquain, 2008; Sarkar, 2008a,b), by showing that there is in fact a family of step-up procedures (depending on the choice of a kind of prior distribution) that control the FDR under unspecified dependence between the p-values.

However, all of these procedures, which are built in order to control the FDR at a level α, can be shown to have actually their FDR upper bounded by $\pi_0\alpha$, where π_0 is the proportion of true null hypotheses in the initial pool. Therefore, when most of the hypotheses are false (i.e., π_0 is small), these procedures are inevitably conservative, since their FDR is in actuality much lower than the fixed target α. In this context, the challenge of *adaptive control* of the FDR (e.g., Benjamini and Hochberg, 2000; Black, 2004) is to integrate an estimation of the unknown proportion π_0 in the threshold of the previous procedures and to prove that the corresponding FDR is still rigorously controlled by α.

Of course, adaptive procedures are of practical interest if it is expected that π_0 is, or can be, significantly smaller than 1. An example of such a situation occurs when using hierarchical procedures (e.g., Benjamini and Heller, 2007) which first selects some clusters of hypotheses that are likely to contain false nulls, and then apply a multiple testing procedure on the selected hypotheses. Since a large part of the true null hypotheses is expected to be false in the second step, an adaptive procedure is needed in order to keep the FDR close to the target level.

A number of adaptive procedures have been proposed in the recent literature and can loosely be divided into the following categories:

- *plug-in* procedures, where some initial estimator of π_0 is directly plugged in as a multiplicative level correction to the usual procedures. In some cases (e.g., Storey's estimator, see Storey, 2002), the resulting plug-in adaptive procedure (or a variation thereof) has been proved to control the FDR at the desired level (Benjamini et al., 2006; Storey et al., 2004). A variety of other estimators of π_0 have been proposed (e.g., Meinshausen and Rice, 2006; Jin and Cai, 2007; Jin, 2008); while their asymptotic consistency (as the number of hypotheses tends to infinity) is generally established, their use in plug-in adaptive procedures has not always been studied.

- *two-stage* procedures: in this approach, a first round of multiple hypothesis testing is performed using some fixed algorithm, then the results of this first round are used in order to tune the parameters of a second round in an adaptive way. This can generally be interpreted as using the output of the first stage to estimate π_0. Different procedures following this general approach have been proposed (Benjamini et al., 2006; Sarkar, 2008a; Farcomeni, 2007); more generally, multiple-stage procedures can be considered.

- *one-stage* procedures, which perform a single round of multiple testing (generally step-up or step-down), based on a particular (deterministic) threshold collection that renders it adaptive (Finner et al., 2009; Gavrilov et al., 2009).

In addition, some works (Genovese and Wasserman, 2004; Storey et al., 2004; Finner et al., 2009) have studied the question of adaptivity to the parameter π_0 from an *asymptotic* viewpoint. In this framework, the more specific *random effects* model is—most generally, though not always—considered, in which p-values are assumed independent, each hypothesis has a probability π_0 of being true, and all false null hypotheses share the same alternate distribution. The behavior of different procedures is then studied under the limit where the number of tested hypotheses grows to infinity. One advantage of this approach and specific model is that it allows to derive quite precise results (see Neuvial, 2008, for a precise study of limiting behaviors of central limit type under this model, including some of the new procedures introduced in the present paper). However, we emphasize that in the present work our focus is decidedly on the nonasymptotic side, using finite samples and arbitrary alternate hypotheses.

To complete this overview, let us also mention another interesting and different direction opened up recently, that of adaptivity to the alternate distribution. If the alternate distributions are known *a priori*, the optimal testing statistics are generally likelihood ratios between each null and each alternate, which (possibly after standardization under the form of p-values) can be combined using a multiple testing algorithm in order to control some measure of type I error while minimizing a measure of type II error (see, e.g., Spjøtvoll, 1972, Wasserman and Roeder, 2006, Genovese et al., 2006, Storey, 2007, Roquain and van de Wiel, 2009). In situations where the alternate is unknown, though, one can hope to estimate, implicitly or explicitly, the alternate distributions from the observed data, and consequently approximate the optimal test statistics and the associated multiple testing procedure (Sun and Cai, 2007 proposed an asymptotically consistent approach to this end). Interestingly, this point of view is also intimately linked to some traditional topics in statistical learning such as classification and of optimal novelty detection (see, e.g., Scott and Blanchard, 2009). However, in the present paper we will focus on adaptivity to the parameter π_0 only.

1.2 Overview of this Paper

The contributions of the present paper are the following. A first goal of the paper is to introduce a number of novel adaptive procedures:

1. We introduce a new *one-stage* step-up procedure that is more powerful than the standard LSU procedure in a large range of situations, and provably controls the FDR under independence (and in a nonasymptotic sense). This procedure is called one-stage adaptive, because the estimation of π_0 is performed implicitly.

2. Based on this, we then build a new *two-stage* adaptive procedure, which is more powerful in general than the procedure proposed by Benjamini et al. (2006), while provably controlling the FDR under independence.

3. Under the assumption of positive or arbitrary dependence of the p-values, we introduce new two-stage adaptive versions of known step-up procedures (namely, of the LSU under positive dependence, and of the family of procedures introduced by Blanchard and Fleuret, 2007, under unspecified dependence). These adaptive versions provably control the FDR and result in an improvement of power over the non-adaptive versions in some situations (namely, when the number of hypotheses rejected in the first stage is large, typically more than 60%).

A second goal of this work is to present a review of several existing adaptive step-up procedures with provable FDR control under independence. For this, we present the theoretical FDR control as a consequence of a single general theorem, which was first established by Benjamini et al. (2006). Here, we give a short self-contained proof of this result that is of independent interest. The latter is based on some tools introduced earlier (Blanchard and Roquain, 2008; Roquain, 2007), aimed at unifying FDR control proofs. Related results and tools also appear independently in Finner et al. (2009) and Sarkar (2008b).

A third goal is to compare both the existing and our new adaptive procedures in an extensive simulation study under both independence and dependence, following the simulation model and methodology used by Benjamini et al. (2006):

- Concerning the new one- and two- stage procedures with theoretical FDR control under independence, these are generally quite competitive in comparison to existing ones. However we also report that the best procedure overall (in terms of power, among procedures that are robust enough to the dependent case) appears to be the plug-in procedure based on the well-known Storey estimator (Storey, 2002) used with the somewhat nonstandard parameter setting $\lambda = \alpha$. This outcome was in part unexpected since to the best of our knowledge, this fact had never been pointed out so far (the usual default recommended choice is $\lambda = \frac{1}{2}$ and turns out to be very unstable in dependent situations); this is therefore an important conclusion of this paper regarding practical use of these procedures.

- Concerning the new two-stage procedures with theoretical FDR control under dependence, simulations show an (admittedly limited) improvement over their non-adaptive counterparts in favorable situations which correspond to what was expected from the theoretical study (i.e., large proportion of false hypotheses). The observed improvement is unfortunately not striking enough to be able to recommend using these procedures in practice.

The paper is organized as follows: in Section 2, we introduce the mathematical framework, and we recall the existing non-adaptive results for FDR control. In Section 3, we deal with the setup of independent p-values. We expose our new procedures and review the existing ones, and compare them theoretically and in a simulation study. The case of positive dependent and arbitrarily dependent p-values is examined in Section 4 where we introduce our new adaptive procedures in this context. A conclusion is given in Section 5. Section 6 and 7 contain proofs of the results and lemmas, respectively. Some technical remarks and discussions of links to other work are gathered at the end of each relevant subsection, and can be skipped by the non-specialist reader.

2. Preliminaries

In this paper, we stick to the traditional statistical framework for multiple testing, which we first briefly recall below.

2.1 Multiple Testing Framework

Let $(\mathcal{X}, \mathfrak{X}, \mathbb{P})$ be a probability space; we aim at inferring a decision on $\mathbb{P}$ from an observation x in $\mathcal{X}$ drawn from $\mathbb{P}$. Let $\mathcal{H}$ be a finite set of null hypotheses for $\mathbb{P}$, that is, each null hypothesis $h \in \mathcal{H}$ corresponds to some subset of distributions on $(\mathcal{X}, \mathfrak{X})$ and "$\mathbb{P}$ satisfies h" means that $\mathbb{P}$ belongs to this subset of distributions. The number of null hypotheses $|\mathcal{H}|$ is denoted by m, where $|.|$ is the cardinality function. The underlying probability $\mathbb{P}$ being fixed, we denote $\mathcal{H}_0 = \{h \in \mathcal{H} \mid \mathbb{P} \text{ satisfies } h\}$ the set of the true null hypotheses and $m_0 = |\mathcal{H}_0|$ the number of true null hypotheses. We let also $\pi_0 := m_0/m$ the proportion of true null hypotheses. We stress that $\mathcal{H}_0$, m_0, and π_0 are unknown and implicitly depend on the unknown $\mathbb{P}$. All the results to come are always implicitly meant to hold for any generating distribution $\mathbb{P}$.

We suppose further that there exists a set of *p-value* functions $\mathbf{p} = (p_h, h \in \mathcal{H})$, meaning that each $p_h : (\mathcal{X}, \mathfrak{X}) \mapsto [0,1]$ is a measurable function and that for each $h \in \mathcal{H}_0$, p_h is bounded stochastically by a uniform distribution, that is,

$$\forall h \in \mathcal{H}_0, \qquad \forall t \in [0,1],\ \mathbb{P}[p_h \leq t] \leq t. \tag{1}$$

Typically, each p-value is obtained from a statistic Z that has a known distribution P_0 under the corresponding null hypothesis. In this case, $p_h = \overline{\Phi}_0(Z)$ satisfies (1) in general, where $\overline{\Phi}_0(z) = P_0([z, +\infty))$. Here, we are however not concerned with how these p-values are precisely constructed and only assume that they exist and are known (this is the standard setting in multiple testing).

2.2 Multiple Testing Procedure and Errors

A *multiple testing procedure* is a function

$$R : x \in \mathcal{X} \mapsto R(x) \in \mathcal{P}(\mathcal{H}),$$

such that for any $h \in \mathcal{H}$, the function $x \mapsto \mathbf{1}\{h \in R(x)\}$ is measurable. It takes as input an observation x and returns a subset of $\mathcal{H}$, corresponding to the rejected hypotheses. As it is commonly the case, we will focus here on multiple testing procedure based on p-values, that is, we will implicitly assume that R is of the form $R(\mathbf{p})$.

A multiple testing procedure R can make two kinds of errors: a *type I error* occurs for h when h is true and is rejected by R, that is, $h \in \mathcal{H}_0 \cap R$. Conversely, *a type II error* occurs for h when h is false and is not rejected by R, that is $h \in \mathcal{H}_0^c \cap R^c$. Following the Neyman-Pearson general philosophy for hypothesis testing, the primary concern is to control the quantity of type I errors of a testing procedure. For this, the most traditional way is to upper bound the "Family-wise error rate" (FWER), which is the probability that one or more true null hypotheses are rejected. However, procedures with a controlled FWER are (by definition) very "cautious" not to make even a single error, and thus reject only few hypotheses. This conservative way of measuring the type I error for multiple hypothesis testing can be a serious hindrance in practice, since it requires to collect large enough data sets so that significant evidence can be found under this strict error control criterion.

More recently, a more liberal measure of type I errors has been introduced in multiple testing (Benjamini and Hochberg, 1995): the *false discovery rate* (FDR), which is the averaged proportion of true null hypotheses in the set of all the rejected hypotheses:

Definition 1 (False discovery rate) *The* false discovery rate *of a multiple testing procedure R for a generating distribution $\mathbb{P}$ is given by*

$$\mathrm{FDR}(R) := \mathbb{E}\left[\frac{|R \cap \mathcal{H}_0|}{|R|}\mathbf{1}\{|R| > 0\}\right]. \tag{2}$$

A classical aim, then, is to build procedures R with FDR upper bounded at a given, fixed level α. Of course, if we choose $R = \emptyset$, meaning that R rejects no hypotheses, trivially $\mathrm{FDR}(R) = 0 \leq \alpha$. Therefore, it is desirable to build procedures R satisfying $\mathrm{FDR}(R) \leq \alpha$ while at the same time having as few type II errors as possible. As a general rule, provided that $\mathrm{FDR}(R) \leq \alpha$, we want to build procedures that reject as many false hypotheses as possible. The absolute power of a multiple testing procedure is defined as the average proportion of false hypotheses correctly rejected, $\mathbb{E}\left[\left|R \cap \mathcal{H}_0^c\right|\right] / \left|\mathcal{H}_0^c\right|$. Given two procedures R and R', a particularly simple sufficient condition for R to be more powerful than R' is when R' if $R' \subset R$ holds pointwise. We will say in this case that R is *(uniformly) less conservative* than R'.

Remark 2 *Throughout this paper we will use the following convention: whenever there is an indicator function inside an expectation, this has logical priority over any other factor appearing in the expectation. What we mean is that if other factors include expressions that may not be defined (such as the ratio $\frac{0}{0}$) outside of the set defined by the indicator, this is safely ignored. This results in more compact notation, such as in Definition 1. Note also again that the dependence of the FDR on the unknown $\mathbb{P}$ is implicit.*

2.3 Self-Consistency, Step-Up Procedures, FDR Control and Adaptivity

We first define a general class of multiple testing procedures called *self-consistent procedures* (Blanchard and Roquain, 2008).

Definition 3 (Self-consistency, nonincreasing procedure) *Let $\Delta : \{0, 1, \ldots, m\} \to \mathbb{R}^+$, $\Delta(0) = 0$, be a nondecreasing function called* threshold collection*; a multiple testing procedure R is said to satisfy the self-consistency condition with respect to Δ if the inclusion*

$$R \subset \{h \in \mathcal{H} \mid p_h \leq \Delta(|R|)\}$$

holds almost surely. Furthermore, we say that R is nonincreasing if for all $h \in \mathcal{H}$ the function $p_h \mapsto |R(\mathbf{p})|$ is nonincreasing, that is if $|R|$ is nonincreasing in each p-value.

The class of self-consistent procedures includes well-known types of procedures, notably step-up and step-down. The assumption that a procedure is nonincreasing, which is required in addition to self-consistency in some of the results to come, is relatively natural as a lower p-value means we have more evidence to reject the corresponding hypothesis. We will mainly focus on the *step-up* procedure, which we define now. For this, we sort the p-values in increasing order using the notation $p_{(1)} \leq \cdots \leq p_{(m)}$ and putting $p_{(0)} = 0$. This order is of course itself random since it depends on the observation.

Definition 4 (Step-up procedure) *The step-up procedure with threshold collection Δ is defined as*

$$R = \{h \in \mathcal{H} \mid p_h \leq p_{(k)}\}, \text{ where } k = \max\{0 \leq i \leq m \mid p_{(i)} \leq \Delta(i)\}.$$

A trivial but important property of a step-up procedure is the following.

Lemma 5 *The step-up procedure with threshold collection Δ is nonincreasing and self-consistent with respect to Δ.*

Therefore, a result valid for any nonincreasing self-consistent procedure w.r.t. Δ holds in particular for the corresponding step-up procedure. This will be used extensively through the paper and thus should be kept in mind by the reader.

Among all procedures that are self-consistent with respect to Δ, the step-up is uniformly less conservative than any other (Blanchard and Roquain, 2008) and is therefore of primary interest. However, to recover procedures of a more general form (including step-down for instance), the statements of this paper will be preferably expressed in terms of self-consistent procedures when it is possible.

Threshold collections are generally scaled by the target FDR level α. Once correspondingly rewritten under the normalized form $\Delta(i) = \alpha\beta(i)/m$, we will call β the *shape function* for threshold collection Δ. In the particular case where the shape function β is the identity function, the procedure is called the *linear step-up* (LSU) *procedure* (at level α).

The LSU plays a prominent role in multiple testing for FDR control; it was the first procedure for which FDR control was proved and it is probably the most widely used procedure in this context. More precisely, when the p-values are assumed to be independent, the following theorem holds.

Theorem 6 *Suppose that the family of p-values $\mathbf{p} = (p_h, h \in \mathcal{H})$ is independent. Then any nonincreasing self-consistent procedure with respect to threshold collection $\Delta(i) = \alpha i/m$ has FDR upper bounded by $\pi_0\alpha$, where $\pi_0 = m_0/m$ is the proportion of true null hypotheses. (In particular, this is the case for the linear step-up procedure.) Moreover, if the p-values associated to true null hypotheses are exactly distributed like a uniform distribution, the linear step-up procedure has FDR exactly equal to $\pi_0\alpha$.*

For the specific case of the LSU, the first part of this result was proved in the landmark paper of Benjamini and Hochberg (1995); the second part was proved by Benjamini and Yekutieli (2001) and Finner and Roters (2001). Benjamini and Yekutieli (2001) extended the first part by proving that the LSU procedure still controls the FDR in the case of p-values with a certain form of positive dependence called *positive regression dependence from a subset* (PRDS). We skip a formal definition for now (we will get back to this topic in Section 4). The extension of these results to self-consistent procedures (in the independent as well as PRDS case) was established by Blanchard and Roquain (2008) and Finner et al. (2009).

However, when no particular assumption is made on the dependence between the p-values, it can be shown that the above FDR control does not hold in general. This situation is called *unspecified* or *arbitrary* dependence. A modification of the LSU was first proposed by Benjamini and Yekutieli (2001), and proved to have a controlled FDR under arbitrary dependence. This result was extended by Blanchard and Fleuret (2007) and Blanchard and Roquain (2008) (see also a related result of Sarkar, 2008a,b). Namely, it can be shown that self-consistent procedures (not necessarily nonincreasing) based on a particular class of shape functions have controlled FDR:

Theorem 7 *Under unspecified dependence of the family of p-values* $\mathbf{p} = (p_h, h \in \mathcal{H})$*, let* β *be a shape function of the form:*

$$\beta(r) = \int_0^r u d\nu(u), \tag{3}$$

where ν *is some fixed a priori probability distribution on* $(0,\infty)$*. Then any self-consistent procedure with respect to threshold collection* $\Delta(i) = \alpha\beta(i)/m$ *has FDR upper bounded by* $\alpha\pi_0$ *.*

To recap, in all of the above cases, the FDR is actually controlled at the level $\pi_0\alpha$ instead of the target α. Hence, a direct corollary of both of the above theorems is that the step-up procedure with shape function $\beta^*(x) = \pi_0^{-1}\beta(x)$ has FDR upper bounded by α in either of the following situations:

- $\beta(x) = x$ when the p-value family is independent or PRDS,
- the shape function β is of the form (3) when the p-values have unspecified dependence.

Since $\pi_0 \leq 1$, using β^* always gives rise to a less conservative procedure than using β (especially when π_0 is small). However, since π_0 is unknown, the shape function β^* is not directly accessible. We therefore call the step-up procedure using β^* the *Oracle step-up procedure* based on shape function β (in each of the above cases).

Simply put, the role of adaptive step-up procedures is to mimic the latter oracle in order to obtain more powerful procedures. Adaptive procedures are often step-up procedures using the modified shape function $G\beta$, where G is some estimator of π_0^{-1}:

Definition 8 (Plug-in adaptive step-up procedure) *Given a level* $\alpha \in (0,1)$*, a shape function* β *and an estimator* $G : [0,1]^{\mathcal{H}} \to (0,\infty)$ *of the quantity* π_0^{-1} *, the plug-in adaptive step-up procedure of shape function* β *and using estimator* G *(at level* α*) is defined as*

$$R = \{h \in \mathcal{H} \mid p_h \leq p_{(k)}\}, \text{ where } k = \max\{0 \leq i \leq m \mid p_{(i)} \leq \alpha\beta(i)G(\mathbf{p})/m\}.$$

The (data-dependent) function $\Delta(\mathbf{p}, i) = \alpha\beta(i)G(\mathbf{p})/m$ *is called the adaptive threshold collection corresponding to the procedure. In the particular case where the shape function* β *is the identity function on* $\mathbb{R}^+$*, the procedure is called an adaptive linear step-up procedure using estimator* G *(and at level* α*).*

Following the previous definition, an adaptive plug-in procedure is composed of two different steps:

1. Estimate π_0^{-1} with an estimator G .
2. Take the step-up procedure of shape function $G\beta$.

A subclass of plug-in adaptive procedures is formed by so-called *two-stage procedures*, when the estimator G is actually based on a first, non-adaptive, multiple testing procedure. This can obviously be possibly iterated and leads to multi-stage procedures. The distinction between generic plug-in procedures and two-stage procedures is somewhat informal and generally meant only to provide some kind of nomenclature between different possible approaches.

The main theoretical task is to ensure that an adaptive procedure of this type still correctly controls the FDR. The mathematical difficulty obviously comes from the additional random variations of the estimator G in the procedure.

3. Adaptive Procedures with Provable FDR Control under Independence

In this section, we introduce two new adaptive procedures that provably control the FDR under independence. The first one is one-stage and does not include an explicit estimator of π_0^{-1}, hence it is not explicitly a plug-in procedure. We then propose to use this as the first stage in a new two-stage procedure, which constitutes the second proposed method.

For clarity, we first introduce the new one-stage procedure; we then discuss several possible plug-in procedures, including our new proposition and several procedures proposed by other authors. FDR control for these various plug-in procedures can be studied under independence using a general theoretical device introduced by Benjamini et al. (2006) which we reproduce here with a self-contained and somewhat simplified proof. Finally, we compare these different approaches; first with a theoretical study of the robustness under a very specific case of maximal dependence; second by extensive simulations, where we inspect both the performance under independence and the robustness under a wide range of positive correlations.

3.1 New Adaptive One-Stage Step-Up Procedure

We present here our first main contribution, a one-stage adaptive step-up procedure. This means that the estimation step is implicitly included in the (deterministic) threshold collection.

Theorem 9 *Suppose that the p-value family* $\mathbf{p} = (p_h, h \in \mathcal{H})$ *is independent and let* $\lambda \in (0,1)$ *be fixed. Define the adaptive threshold collection*

$$\Delta(i) = \min\left((1-\lambda)\frac{\alpha i}{m-i+1}, \lambda\right). \tag{4}$$

Then any nonincreasing self-consistent procedure with respect to Δ *has FDR upper bounded by* α. *In particular, this is the case of the corresponding step-up procedure, denoted by BR-1S-*λ.

The above result is proved in Section 6. Our proof is in part based on Lemma 1 of Benjamini et al. (2006). Note that an alternate proof of Theorem 9 is established in Sarkar (2008b) without using this lemma, while nicely connecting the FDR upper-bound to the false non-discovery rate.

3.1.1 COMPARISON TO THE LSU

Below, we will mainly focus on the choice $\lambda = \alpha$, leading to the threshold collection

$$\Delta(i) = \alpha \min\left((1-\alpha)\frac{i}{m-i+1}, 1\right). \tag{5}$$

For $i \leq (m+1)/2$, the threshold (5) is $\alpha\frac{(1-\alpha)i}{m-i+1}$, and thus our approach differs from the threshold collection of the standard LSU procedure threshold by the factor $\frac{(1-\alpha)m}{m-i+1}$.

It is interesting to note that the correction factor $\frac{m}{m-i+1}$ appears in Holm's step-down procedure (Holm, 1979) for FWER control. The latter is a well-known improvement of Bonferroni's procedure (which corresponds to the fixed threshold α/m), taking into account the proportion of true nulls, and defined as the step-down procedure[1] with threshold collection $\alpha/(m-i+1)$. Here we therefore

1. The step-down procedure with threshold collection Δ rejects the hypotheses corresponding to the k smallest p-values, where $k = \max\{0 \leq i \leq m \mid \forall j \leq i,\ p_{(j)} \leq \Delta(j)\}$. It is self-consistent with respect to Δ but uniformly more conservative than the step-up procedure with the same threshold collection, compare with Definition 4.

prove that this correction is suitable as well for the linear step-up procedure, in the framework of FDR control.

If r denotes the final number of rejections of the new one-stage procedure, we can interpret the ratio $\frac{(1-\alpha)m}{m-r+1}$ between the adaptive threshold and the LSU threshold at the same point as an *a posteriori* estimate for π_0^{-1} . In the next section we propose to use this quantity in a plug-in, two-stage adaptive procedure.

As Figure 1 illustrates, our procedure is generally less conservative than the (non-adaptive) linear step-up procedure (LSU). Precisely, the new procedure can only be more conservative than the LSU procedure in the marginal case where the factor $\frac{(1-\alpha)m}{m-i+1}$ is smaller than one. This happens only when the proportion of null hypotheses rejected by the LSU procedure is positive but less than $\alpha + 1/m$ (and even in this region the ratio of the two threshold collections is never less than $(1-\alpha)$). Roughly speaking, this situation with only few rejections can only happen if there are few false hypotheses to begin with (π_0 close to 1) or if the false hypotheses are very difficult to detect (the distribution of false p-values is close to being uniform).

In the interest of being more specific, we briefly investigate this issue in the next lemma, considering the particular *Gaussian random effects* model (which is relatively standard in the multiple testing literature, see, for example, Genovese and Wasserman, 2004) in order to give a quantitative answer from an asymptotical point of view (when the number of tested hypotheses grows to infinity). In the random effect model, hypotheses are assumed to be randomly true or false with probability π_0 , and the false null hypotheses share a common distribution P_1 . Globally, the p-values then are i.i.d. drawn according to the mixture distribution $\pi_0 U[0,1] + (1-\pi_0)P_1$.

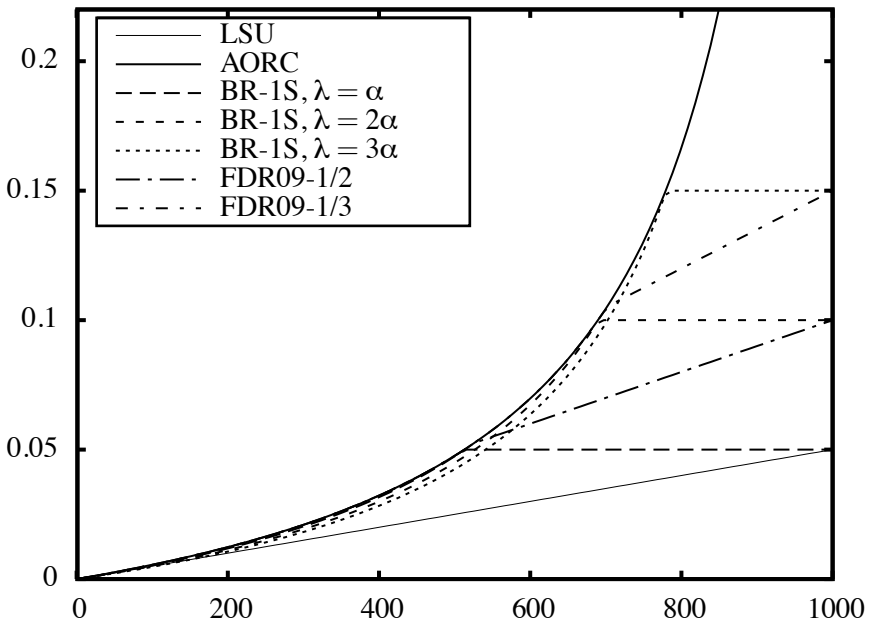


Figure 1: For $m = 1000$ null hypotheses and $\alpha = 5\%$: comparison of the new threshold collection *BR-1S*-λ given by (4) to that of the LSU, the AORC and *FDR09*-η .

Lemma 10 *Consider the random effects model where the p-values are i.i.d. with common cumulative distribution function* $t \mapsto \pi_0 t + (1-\pi_0)F(t)$. *Assume that the true null hypotheses are standard Gaussian with zero mean and that the alternative hypotheses are standard Gaussian with mean* $\mu > 0$. *In this case* $F(t) = \overline{\Phi}(\overline{\Phi}^{-1}(t) - \mu)$, *where* $\overline{\Phi}$ *is the standard Gaussian upper tail function. Assuming* $\pi_0 < (1+\alpha)^{-1}$, *define*

$$\mu^\star = \overline{\Phi}^{-1}(\alpha^2) - \overline{\Phi}^{-1}\left(\frac{\alpha^{-1}-\pi_0}{1-\pi_0}\alpha^2\right).$$

Then if $\mu > \mu^*$, *the probability that the LSU rejects a proportion of null hypotheses less than* $1/m+\alpha$ *tends to 0 as m tends to infinity. On the other hand, if* $\pi_0 > (1+\alpha)^{-1}$, *or* $\mu < \mu^*$, *then this probability tends to one.*

Lemma 10 is proved in Section 6. Taking for instance in this lemma the values $\pi_0 = 0.5$ and $\alpha = 0.05$, results in the critical value $\mu^\star \simeq 1.51$. This lemma delineates clearly in a particular case in which situation we can expect an improvement from the adaptive procedure BR-1S over the standard LSU.

3.1.2 COMPARISON TO OTHER ADAPTIVE ONE-STAGE PROCEDURES

Very recently, other adaptive one-stage procedures with important similarities to *BR-1S-*λ have been proposed by other authors. (The present work was developed independently.)

Starting with some heuristic motivations, Finner et al. (2009) proposed the threshold collection $t(i) = \frac{\alpha i}{m-(1-\alpha)i}$, which they dubbed the *asymptotically optimal rejection curve* (AORC). However, the step-up procedure using this threshold collection as is does not have controlled FDR (since $t(m) = 1$, the corresponding step-up procedure would always reject all the hypotheses), and several suitable modifications were proposed by Finner et al. (2009), the simplest one being

$$t'_\eta(i) = \min\left(t(i), \eta^{-1}\alpha i/m\right),$$

which is denoted by *FDR09-*η in the following.

The theoretical FDR control proved in Finner et al. (2009) is studied asymptotically as the number of hypotheses grows to infinity. In that framework, asymptotical control at level α is shown to hold for any $\eta < 1$. On Figure 1, we represented the thresholds *BR-1S-*λ and *FDR09-*η for comparison, for several choices of the parameters. The two families appear quite similar, initially following the AORC curve, then branching out or capping at a point depending on the parameter. One noticeable difference in the initial part of the curve is that while *FDR09-*η exactly coincides with the AORC, *BR-1S-*λ is arguably sligthly more conservative. This reflects the nature of the corresponding theoretical result—nonasymptotic control of the FDR requires a somewhat more conservative threshold as compared to the only asymptotic control of *FDR-*η . Additionally, we can use *BR-1S-*λ as a first step in a 2-step procedure, as will be argued in the next section.

The ratio between *BR-1S-*λ and the AORC (before the capping point) is a factor which, assuming $\alpha \geq (m+1)^{-1}$, is lower bounded by $(1-\lambda)(1-\frac{1}{m+1})$. This suggests that the value for λ should be kept small, this is why we propose $\lambda = \alpha$ as a default choice.

Finally, the *step-down* procedure based on the AORC threshold collection (under the slightly modified form $\widetilde{t}(i) = \frac{\alpha i}{m-(1-\alpha)i+1}$, but with no further modification) is proposed and studied by Gavrilov et al. (2009). Using specific properties of step-down procedures, these authors proved the nonasymptotic FDR control of this procedure.

3.2 Adaptive Plug-In Methods

In this section, we consider different adaptive step-up procedures of the plug-in type, that is, based on an explicit estimator of π_0^{-1}. We first review a general method proposed by Benjamini et al. (2006) in order to derive FDR control for such plug-in procedures (see also Theorem 4.3 of Finner et al., 2009, for a similar result, as well as Theorem 3.3 of Sarkar, 2008b). We propose here a self-contained proof of this result, which notably extends the original result from step-up procedures to more general self-consistent procedures. Based on this result, we review the different plug-in estimators considered by Benjamini et al. (2006) and add a new one to the lot, based on the one-stage adaptive procedure introduced in the previous section.

Let us first introduce the following notation: for each $h \in \mathcal{H}$, we denote by $\mathbf{p}_{-h}$ the collection of p-values $\mathbf{p}$ restricted to $\mathcal{H} \setminus \{h\}$, that is, $\mathbf{p}_{-h} = (p_{h'}, h' \neq h)$. We also denote $\mathbf{p}_{0,h} = (\mathbf{p}_{-h}, 0)$ the collection $\mathbf{p}$ where p_h has been replaced by 0.

Theorem 11 (Benjamini, Krieger, Yekutieli 2006) *Suppose that the p-value family $\mathbf{p} = (p_h, h \in \mathcal{H})$ is independent. Let $G : [0,1]^{\mathcal{H}} \to (0,\infty)$ be a measurable, coordinate-wise nonincreasing function. Consider a nonincreasing multiple testing procedure R which is self-consistent with respect to the adaptive linear threshold collection $\Delta(\mathbf{p}, i) = \alpha G(\mathbf{p}) i/m$. Then the following holds:*

$$\mathrm{FDR}(R) \leq \frac{\alpha}{m} \sum_{h \in \mathcal{H}_0} \mathbb{E}\left[G(\mathbf{p}_{0,h})\right]. \tag{6}$$

In particular, if for any $h \in \mathcal{H}_0$, it holds that $\mathbb{E}\left[G(\mathbf{p}_{0,h})\right] \leq \pi_0^{-1}$, then $\mathrm{FDR}(R) \leq \alpha$.

The proof is given in Section 6. Since we assumed G to be nonincreasing, the quantity $\mathbb{E}\left[G(\mathbf{p}_{0,h})\right]$ in bound (6) is maximized when the p-values associated to true nulls have a uniform distribution (p_h excepted), while the p-values associated to false nulls are all set to zero. Following Finner et al. (2009), this *least favorable configuration* for the distribution of p-values is referred to as the Dirac-Uniform distribution and gives rise to the following corollary:

Corollary 12 *Consider the same conditions as for Theorem 11, and assume moreover that G is invariant by permutation of the p-values. Then it holds that*

$$\mathrm{FDR}(R) \leq \gamma(G,m)\,\alpha,$$

with $\gamma(G,m) = \max_{1 \leq m_0 \leq m} \left\{ \frac{m_0}{m} \mathbb{E}_{\mathbf{p} \sim DU(m, m_0 - 1)}\left[G(\mathbf{p})\right] \right\}$, where $DU(m,j)$ is the distribution of $\mathbf{p}$ where the j first p-values are independent uniform in $[0,1]$ and the $m-j$ others are identically equal to zero.

(While the proof is standard, it is given for completeness in Section 6). The interest of the last result is that for *any* choice of nonincreasing (permutation invariant) function G, it is possible in principle to evaluate $\gamma(G,m)$ by a Monte Carlo method, namely by estimating the expected value of G under the $m-1$ possible least favorable configurations. This leads to a practical control of the FDR valid for any value of m_0, obtained by dividing the target level α by $\gamma(G,m)$ before applying the procedure.

However, when m is large, this method can be computationally demanding, and a more convenient approach for practical use is to obtain explicit bounds for specific estimators. We now

concentrate on this goal and apply the result of Theorem 11 (or alternatively of Corollary 12) to the following estimators, depending on a fixed parameter $\lambda \in (0,1)$ or $k_0 \in \{1,\dots,m\}$:

$$\text{[Storey-}\lambda\text{]} \qquad G_1(\mathbf{p}) = \frac{(1-\lambda)m}{\sum_{h\in\mathcal{H}} \mathbf{1}\{p_h > \lambda\}+1};$$

$$\text{[Quant-}\frac{k_0}{m}\text{]} \qquad G_2(\mathbf{p}) = \frac{(1-p_{(k_0)})m}{m-k_0+1};$$

$$\text{[BKY06-}\lambda\text{]} \qquad G_3(\mathbf{p}) = \frac{(1-\lambda)m}{m-|R_0(\mathbf{p})|+1}, \text{ where } R_0 \text{ is the standard LSU at level } \lambda;$$

$$\text{[BR-2S-}\lambda\text{]} \qquad G_4(\mathbf{p}) = \frac{(1-\lambda)m}{m-|R'_0(\mathbf{p})|+1}, \text{ where } R'_0 \text{ is BR-1S-}\lambda \text{ (see Theorem 9).}$$

Above, the notation "Storey-λ", "Quant-$\frac{k_0}{m}$", "BKY06-λ" and "BR-2S-λ" refer to the plug-in adaptive linear step-up procedures associated to G_1, G_2, G_3 and G_4, respectively.

Estimator G_1 is usally called *modified Storey's estimator* and was initially introduced by Storey (2002) from an heuristics on the p-values histogram (originally without the "+1", hence the name "modified"). Its intuitive justification is as follows: denoting by S_λ the set of p-values larger than the threshold λ, the average number of true nulls having a p-value in S_λ is $m_0(1-\lambda)$. Hence, a natural estimator of π_0^{-1} is $(1-\lambda)m/|S_\lambda \cap \mathcal{H}_0| \geq (1-\lambda)m/|S_\lambda| \simeq G_1(\mathbf{p})$. In particular, we expect that Storey's estimator is generally an underestimate of π_0^{-1}, which is in accordance with the condition of Theorem 11. A standard choice is $\lambda = 1/2$ (as in the SAM software of Storey and Tibshirani, 2003). FDR control for the corresponding plug-in step-up procedure was proved by Storey et al. (2004) (more precisely, for the modification $\widetilde{\Delta}(\mathbf{p},i) = \min(\alpha G_1(\mathbf{p})i/m, \lambda)$) and by Benjamini et al. (2006).

Estimator G_2 was introduced by Benjamini and Hochberg (2000) and Efron et al. (2001), from a slope heuristics on the p-values c.d.f. Roughly speaking, G_2 appears as Storey's estimator with the data-dependent parameter choice $\lambda = p_{(k_0)}$, and can therefore be interpreted as the quantile version of Storey's estimator. A standard value for k_0 is $\lfloor m/2 \rfloor$, resulting in the so-called median adaptive LSU (see Benjamini et al., 2006, and the references therein).

Estimator G_3 was introduced by Benjamini et al. (2006) for the particular choice $\lambda = \alpha/(1+\alpha)$. More precisely, a slightly less conservative version, without the "+1" in the denominator, was used in Benjamini et al. (2006). We forget about this refinement here, noting that it results only in a very slight improvement.

Finally, the estimator G_4 is new and follows exactly the same philosophy as G_3, that is, uses a step-up procedure as a first stage in order to estimate π_0^{-1}, but this time based on our adaptive one-stage step-up procedure introduced in the previous section, rather than the standard LSU. Note that since R'_0 is less conservative than R_0 (except in marginal cases), we generally have $G_3 \leq G_4$ pointwise and our estimator improves over the one of Benjamini et al. (2006).

These different estimators all satisfy the sufficient condition mentioned in Theorem 11, and we thus obtain the following corollary:

Corollary 13 *Assume that the family of p-values* $\mathbf{p} = (p_h, h \in \mathcal{H})$ *is independent. For* $i = 1,2,3,4$, *and any* $h \in \mathcal{H}_0$, *it holds that* $\mathbb{E}[G_i(\mathbf{p}_{0,h})] \leq \pi_0^{-1}$. *Therefore, the plug-in adaptive linear step-up procedure at level* α *using estimator* G_i *has FDR smaller than or equal to* α.

The above result for G_1, G_2 and G_3 (for the specific parameter setting $\lambda = \alpha/(1+\alpha)$) was proved by Benjamini et al. (2006). In Section 6, we shortly reproduce their arguments, and prove the result for G_4.

To sum up, Corollary 13 states that under independence, for any λ and k_0, the plug-in adaptive procedures Storey-λ, Quant-$\frac{k_0}{m}$, BKY06-λ and BR-2S-λ all control the FDR at level α.

Remark 14 *The result proved by Benjamini et al. (2006) is actually slightly sharper than Theorem 11. Namely, if $G(\cdot)$ is moreover supposed to be coordinate-wise left-continuous, it is possible to prove that Theorem 11 still holds when $\mathbf{p}_{0,h}$ in the RHS of* (6) *is replaced by the slightly better $\widetilde{\mathbf{p}}_h = (\mathbf{p}_{-h}, \widetilde{p}_h(\mathbf{p}_{-h}))$, defined as the collection of p-values $\mathbf{p}$ where p_h has been replaced by $\widetilde{p}_h(\mathbf{p}_{-h}) = \max\left\{p \in [0,1] \;\middle|\; p \leq \alpha |R(\mathbf{p}_{-h}, p)| G(\mathbf{p}_{-h}, p)\right\}$. This improvement then permits to get rid of the "+1" in the denominator of G_3. Here, we opted for simplicity and a more straightforward statement, noting that this improvement is not crucial.*

Remark 15 *The one-stage step-up procedure of Finner et al. (2009) (see previous discussion in Section 3.1.2)—for which there is no result proving nonasymptotic FDR control up to our knowledge—can also be interpreted intuitively as an adaptive version of the LSU using estimator G_2, where the choice of parameter k_0 is data-dependent. Namely, assume that we want to reject at least i null hypotheses whenever $p_{(i)}$ is lower than the standard LSU threshold times the estimator G_2 wherein parameter $k_0 = i$ is used. This corresponds to the inequality $p_{(i)} \leq \frac{k(1-p_{(i)})}{m-i+1}$, which, solved in $p_{(i)}$, gives the threshold collection of Finner et al. (2009). Remember from Section 3.1.2 that this threshold collection must actually be modified in order to be useful, since it otherwise always leads to reject all hypotheses. The modification leading to FDR09-η consists in capping the estimated π_0^{-1} at a level η, that is, using $\min(\eta, G_2)$ instead of G_2 in the above reasoning. In fact, the proof of Finner et al. (2009) relies on a result which is essentially a reformulation of Theorem 11 for a specific form of estimator.*

Remark 16 *The estimators G_i, $i = 1,2,3,4$ are not necessarily larger than 1, and to this extent can in some unfavorable cases result in the final procedure being actually more conservative than the standard LSU. This can only happen in the situation where either π_0 is close to 1 ("sparse signal") or the alternative hypotheses are difficult to detect ("weak signal"); if such a situation is anticipated, it is more appropriate to use the regular non-adaptive LSU.*

For the Storey-λ estimator, we can control precisely the probability that such an unfavorable case arises by using Hoeffding's inequality (Hoeffding, 1963): assuming the true nulls are i.i.d. uniform on $(0,1)$ and the false nulls i.i.d. of c.d.f. $F(\cdot)$, we write by definition of G_1

$$\begin{aligned}\mathbb{P}[G_1(\mathbf{p}) < 1] &= \mathbb{P}\left[\frac{1}{m}\sum_{h\in\mathcal{H}}^{m}(\mathbf{1}\{p_h > \lambda\} - \mathbb{P}[p_h > \lambda]) > (1-\pi_0)(F(\lambda)-\lambda) - m^{-1}\right] \\ &\leq \exp(-2(mc^2+1)),\end{aligned}$$

where we denoted $c = (1-\pi_0)(F(\lambda)-\lambda)$, and assumed additionally $c > m^{-1}$. The behavior of the bound mainly depends on c, which can get small only if π_0 is close to 1 (sparse signal) or $F(\lambda)$ is close to λ (weak signal), illustrating the above point. In general, provided $c > 0$ does not depend on m, the probability that the Storey procedure fails to outperform the LSU vanishes exponentially as m tends to infinity.

3.3 Theoretical Robustness of the Adaptive Procedures under Maximal Dependence

For the different procedures proposed above, the theory only provides the correct FDR control under independence between the p-values. An important issue is to know how robust this control is when dependence is present (as it is often the case in practice). However, the analytic computation of the FDR under dependence is generally a difficult task, and this issue is often tackled empirically through simulations in a pre-specified model (we will do so in Section 3.4).

In this short section, we present theoretical computations of the FDR for the previously introduced adaptive step-up procedures, under the maximally dependent model where all the p-values are in fact equal, that is $p_h \equiv p_1$ for all $h \in \mathcal{H}$ (and $m_0 = m$). It corresponds to the case where we perform m times the same test, with the same p-value. Albeit relatively trivial and limited, this case leads to very simple FDR computations and provides at least some hints concerning the robustness under dependence of the different procedures studied above.

Proposition 17 *Suppose that we observe m identical p-values $\mathbf{p} = (p_1, ..., p_m) = (p_1, ..., p_1)$ with $p_1 \sim U([0,1])$ and assume $m = m_0$. Then, the following holds:*

$$\begin{aligned}
\mathrm{FDR}(\textit{BR-1S-}\lambda) &= \min\big(\lambda, \alpha(1-\lambda)m\big),\\
\mathrm{FDR}(\textit{FDR09-}\eta) &= \alpha\eta^{-1},\\
\mathrm{FDR}(\textit{Storey-}\lambda) &= \min\big(\lambda, \alpha(1-\lambda)m\big) + \big(\alpha(1-\lambda)(1+m^{-1}) - \lambda\big)_+,\\
\mathrm{FDR}(\textit{Quant-}k_0/m) &= \frac{\alpha}{(1+\alpha) - (k_0-1)m^{-1}},\\
\mathrm{FDR}(\textit{BKY06-}\lambda) &= \mathrm{FDR}(\textit{BR-2S-}\lambda) = \mathrm{FDR}(\textit{Storey-}\lambda).
\end{aligned}$$

Interestingly, the above proposition suggests specific choices of the parameters λ, η and k_0 to ensure control of the FDR at level α under maximal dependence:

- For BR-1S-λ, putting $\lambda_2 = \alpha/(\alpha + m^{-1})$, Proposition 17 gives that $\mathrm{FDR}(\textit{BR-1S-}\lambda) = \lambda$ whenever $\lambda \leq \lambda_2$. This suggests to take $\lambda = \alpha$, and is thus in accordance with the default choice proposed in Section 3.1.

- For FDR09-η, no choice of $\eta < 1$ will lead to the correct FDR control under maximal dependence. However, the larger η , the smaller the FDR in this situation. Note that $\mathrm{FDR}(\textit{FDR09-}\frac{1}{2}) = 2\alpha$.

- For Storey-λ, BKY06-λ and BR-2S-λ, putting $\lambda_1 = \alpha/(1+\alpha+m^{-1})$, we have FDR $= \lambda$ for $\lambda_1 \leq \lambda \leq \lambda_2$. This suggests to choose $\lambda = \alpha$ within these three procedures. Furthermore, note that the standard choice $\lambda = 1/2$ for Storey-λ leads to a very poor control under maximal dependence: $\mathrm{FDR}(\textit{Storey-}\frac{1}{2}) = \min(\alpha m, 1)/2$.

- For Quant-k_0/m, we see that the value of k_0 maximizing the FDR while maintaining it below α is $k_0 = \lfloor \alpha m \rfloor + 1$. Remark also that the standard choice $k_0 = \lfloor m/2 \rfloor$ leads to $\mathrm{FDR}(\textit{Quant-}k_0/m) = 2\alpha/(1+2\alpha+2m^{-1}) \simeq 2\alpha$.

Nevertheless, we would like to underline that the above computations should be interpreted with caution, as the maximal dependence case is very specific and cannot possibly give an accurate idea of the behavior of the different procedures when the correlation between the p-values are strong

but not equal to 1 . For instance, it is well-known that the LSU procedure has FDR far below α for strong positive correlations, but its FDR is equal to α in the above extreme model (see Finner et al., 2007, for a comprehensive study of the LSU under positive dependence). Conversely, the FDR of some adaptive procedures can be higher under moderate dependence than under maximal dependence. This behavior appears in the simulations of the next section, illustrating the complexity of the issue.

3.4 Simulation Study

How can we compare the different adaptive procedures defined above? For a fixed λ, it holds pointwise that $G_1 \geq G_4 \geq G_3$, which shows that the adaptive procedure [Storey-λ] is always less conservative than [BR-2S-λ], itself less conservative than [BKY06-λ] (except in the marginal cases where the one-stage adaptive procedure is more conservative than the standard step-up procedure, as delineated earlier for example in Lemma 10). It would therefore appear that one should always choose [Storey-λ] and disregard the other ones. However, an important point made by Benjamini et al. (2006) for introducing G_3 as a better alternative to the (already known earlier) G_1 is that, on simulations with positively dependent test statistics, the plug-in procedure using G_1 with $\lambda = 1/2$ had very poor control of the FDR, while the FDR was still controlled for the plug-in procedure based on G_3. While the positively dependent case is not covered by the theory, it is of course very important to ensure that a multiple testing procedure is sufficiently robust in practice so that the FDR does not vary too much in this situation.

In order to assess the quality of our new procedures, we compare here the different methods on a simulation study following the setting used by Benjamini et al. (2006). Let $X_i = \mu_i + \varepsilon_i$, for $i, 1 \leq i \leq m$, where ε is a $\mathbb{R}^m$-valued centred Gaussian random vector such that $\mathbb{E}(\varepsilon_i^2) = 1$ and for $i \neq j$, $\mathbb{E}(\varepsilon_i \varepsilon_j) = \rho$, where $\rho \in [0,1]$ is a correlation parameter. Thus, when $\rho = 0$ the X_i's are independent, whereas when $\rho > 0$ the X_i's are positively correlated (with a constant pairwise correlation). For instance, the ε_i's can be constructed by taking $\varepsilon_i := \sqrt{\rho}\, U + \sqrt{1-\rho}\, Z_i$, where Z_i, $1 \leq i \leq m$ and U are all i.i.d $\sim \mathcal{N}(0,1)$.

Considering the one-sided null hypotheses h_i : "$\mu_i \leq 0$" against the alternatives "$\mu_i > 0$" for $1 \leq i \leq m$, we define the p-values $p_i = \overline{\Phi}(X_i)$, for $1 \leq i \leq m$, where $\overline{\Phi}$ is the standard Gaussian distribution tail. We choose a common mean $\bar{\mu}$ for all false hypotheses, that is, for $i, 1 \leq i \leq m_0$, $\mu_i = 0$ and for $i, m_0 + 1 \leq i \leq m, \mu_i = \bar{\mu}$; the p-values corresponding to the null means follow exactly a uniform distribution.

Note that the case $\rho = 1$ and $m = m_0$ (i.e., $\pi_0 = 1$) corresponds to the maximally dependent case studied in Section 3.3.

We compare the following step-up multiple testing procedures: first, the one-stage step-up procedures defined in Section 3.1:

- [BR08-1S-α] The new procedure of Theorem 9, with parameter $\lambda = \alpha$,
- [FDR09-$\frac{1}{2}$] The procedure proposed in Finner et al. (2009) and described in Section 3.1.2, with $\eta = \frac{1}{2}$.

Secondly, the adaptive plug-in step-up procedures defined in Section 3.2:

- [Median LSU] The procedure [Quant-$\frac{k_0}{m}$] with the choice $\frac{k_0}{m} = \frac{1}{2}$,
- [BKY06-α] The procedure [BKY06-λ] with the parameter choice $\lambda = \alpha$,

- [BR08-2S-α] The procedure [BR08-2S-λ] with the parameter choice $\lambda = \alpha$,
- [Storey-λ] With the choices $\lambda = 1/2$ and $\lambda = \alpha$.

Finally, we used as oracle reference [LSU Oracle], the step-up procedure with the threshold collection $\Delta(i) = \alpha i/m_0$, using "oracle" prior knowledge of π_0.

The parameter choice $\lambda = \alpha$ for [Storey-λ] comes from the relationship (delineated in Section 3.1) of G_3, G_4 to G_1, and from the discussion of the maximally dependent case in Section 3.3. Note that the procedure studied by Benjamini et al. (2006) is actually [BKY06-$\alpha/(1+\alpha)$] in our notation (up to a minor modification explained in Remark 14). Thefore, the procedure [BKY06-α] used in our simulations is not srictly the same as in Benjamini et al. (2006), but it is very close.

The three most important parameters in the simulation are the correlation coefficient ρ, the proportion of true null hypotheses π_0, and the alternative mean $\bar{\mu}$ which represents the signal-to-noise ratio, or how easy it is to distinguish alternative hypotheses. We present in Figures 2, 3, and 4 results of the simulations for one varying parameter (π_0, $\bar{\mu}$ and ρ, respectively), the others being kept fixed. Reported are, for the different methods: the FDR, and the power relative to the reference [LSU-Oracle]. Remember the absolute power is defined as the mean proportion of false null hypotheses that are correctly rejected; for each procedure the relative power is the ratio of its absolute power to that of [LSU-Oracle]. Each point is estimated by an average of 10^5 simulations, with fixed parameters $m = 100$ and $\alpha = 5\%$.

3.4.1 UNDER INDEPENDENCE ($\rho = 0$)

Remember that under independence of the p-values, the procedure [LSU] has a FDR equal to $\alpha\pi_0$ and that the procedure [LSU Oracle] has a FDR equal to α (provided that $\alpha \leq \pi_0$). The other procedures have their FDR upper bounded by α (in an asymptotical sense only for [FDR09-$\frac{1}{2}$]).

The situation where the p-values are independent corresponds to the first row of Figures 2 and 3 and the leftmost point of each graph in Figure 4. It appears that in the independent case, the following procedures can be consistently ordered in terms of (relative) power over the range of parameters studied here:

$$[\text{Storey-}\tfrac{1}{2}] \succ [\text{Storey-}\alpha] \succ [\text{BR08-2S-}\alpha] \succ [\text{BKY06-}\alpha],$$

the symbol "$\succ$" meaning "is (uniformly over our experiments) more powerful than".

Next, the procedures [median-LSU] and [FDR09-$\frac{1}{2}$] appear both consistently less powerful than [Storey-$\frac{1}{2}$], and [FDR09-$\frac{1}{2}$] is additionally also consistently less powerful than [Storey-α]. Their relation to the remaining procedures depends on the parameters; both [median-LSU] and [FDR09-$\frac{1}{2}$] appear to be more powerful than the remaining procedures when $\pi_0 > \frac{1}{2}$, and less efficient otherwise. We note that [median-LSU] also appears to perform better when $\bar{\mu}$ is low (i.e., the alternative hypotheses are harder to distinguish).

Concerning our one-stage procedure [BR08-1S-α], we note that it appears to be indistinguishable from its two-stage counterpart [BR08-2S-α] when $\pi_0 > \frac{1}{2}$, and significantly less powerful otherwise. This also corresponds to our expectations, since in the situation $\pi_0 < \frac{1}{2}$, there is a much higher likelihood that more than 50% hypotheses are rejected, in which case our one-stage threshold family hits its "cap" at level α (see, e.g., Fig. 1; a similar qualitative explanation applies to understand the behavior of [FDR09-$\frac{1}{2}$]). This is precisely to improve on this situation that we introduced the two-stage procedure, and we see that the latter does in fact improve substantially the one-stage version in that specific region.

The fact that [Storey-$\frac{1}{2}$] is uniformly more powerful than the other procedures in the independent case corroborates the simulations reported in Benjamini et al. (2006). Generally speaking, under independence we obtain a less biased estimate for π_0^{-1} when considering Storey's estimator based on a "high" threshold like $\lambda = \frac{1}{2}$. Namely, higher p-values are less likely to be "contaminated" by false null hypotheses; conversely, if we take a lower threshold λ, there will be more false null hypotheses included in the set of p-values larger than λ , leading to a pessimistic bias in the estimation of π_0^{-1} . This qualitative reasoning is also consistent with the observed behavior of [median-LSU], since the set of p-values larger than the median is much more likely to be "contaminated" when $\pi_0 < \frac{1}{2}$.

However, the problem with [Storey-$\frac{1}{2}$] is that the corresponding estimation of π_0^{-1} exhibits much more variability than its competitors when there is a substantial correlation between the p-values. As a consequence it is a very fragile procedure. This phenomenon was already pinpointed in Benjamini et al. (2006) and we study it next.

3.4.2 UNDER POSITIVE DEPENDENCE ($\rho > 0$)

Under positive dependence, remember that it is known theoretically from Benjamini and Yekutieli (2001) that the FDR of the procedure [LSU] (resp. [LSU Oracle]) is still bounded by $\alpha\pi_0$ (resp. α), but without equality in general. However, we do not know from a theoretical point of view if the adaptive procedures have their FDR upper bounded by α. In fact, it was pointed out by Farcomeni (2007), in another work reporting simulations on adaptive procedures, that one crucial point to this respect seems to be the variability of estimate of π_0^{-1}. Estimates of this quantity that are not robust with respect to positive dependence will result in failures for the corresponding multiple testing procedure.

The situation where the p-values are positively dependent corresponds to the second and third rows ($\rho = 0.2, 0.5$, respectively) of Figures 2 and 3 and to all the graphs of Figure 4 (except the leftmost points corresponding to $\rho = 0$).

The most striking fact is that [Storey-$\frac{1}{2}$] does not control the FDR at the desired level any longer under positive dependence, and can even be off by quite a large factor. This is in accordance with the experimental findings of Benjamini et al. (2006). Therefore, although this procedure was the favorite in the independent case, it turns out to be not robust, which is very undesirable for practical use where it is generally impossible to guarantee that the p-values are independent. The procedure [median-LSU] appears to have higher power than the remaining ones in the situations studied in Figure 3, especially with a low signal-to-noise ratio. Unfortunately, other situations appearing in Figures 2 and 4 show that [median-LSU] can exhibit a poor FDR control in some parameter regions, most notably when π_0 is close to 1 and positive dependence is present (see, e.g., Figure 4, bottom row). In a majority of practical situations, this is an important drawback since it is difficult to rule out *a priori* that π_0 is close to 1 (i.e., there is only a small proportion of false hypotheses), or that dependence is present. Additionally, from the inspection of the behavior of the power of [median-LSU] in Figures 2 and 4, it appears that the parameter setting $\pi_0 = 0.5$ (which is the fixed value used in Figure 3) is actually noticeably the most favorable for [median-LSU] under dependence. For other values of π_0, this procedure is often clearly outperformed in terms of power, in particular by [Storey-α] and [BR-2S-α]. (At this point we have no satisfying explanation to this peculiar "peak of power" at $\pi_0 = 0.5$ observed specifically for the [median-LSU] procedure under dependence.) For all of these reasons, our conclusion is that [median-LSU] is also not robust enough in general to be reliable.

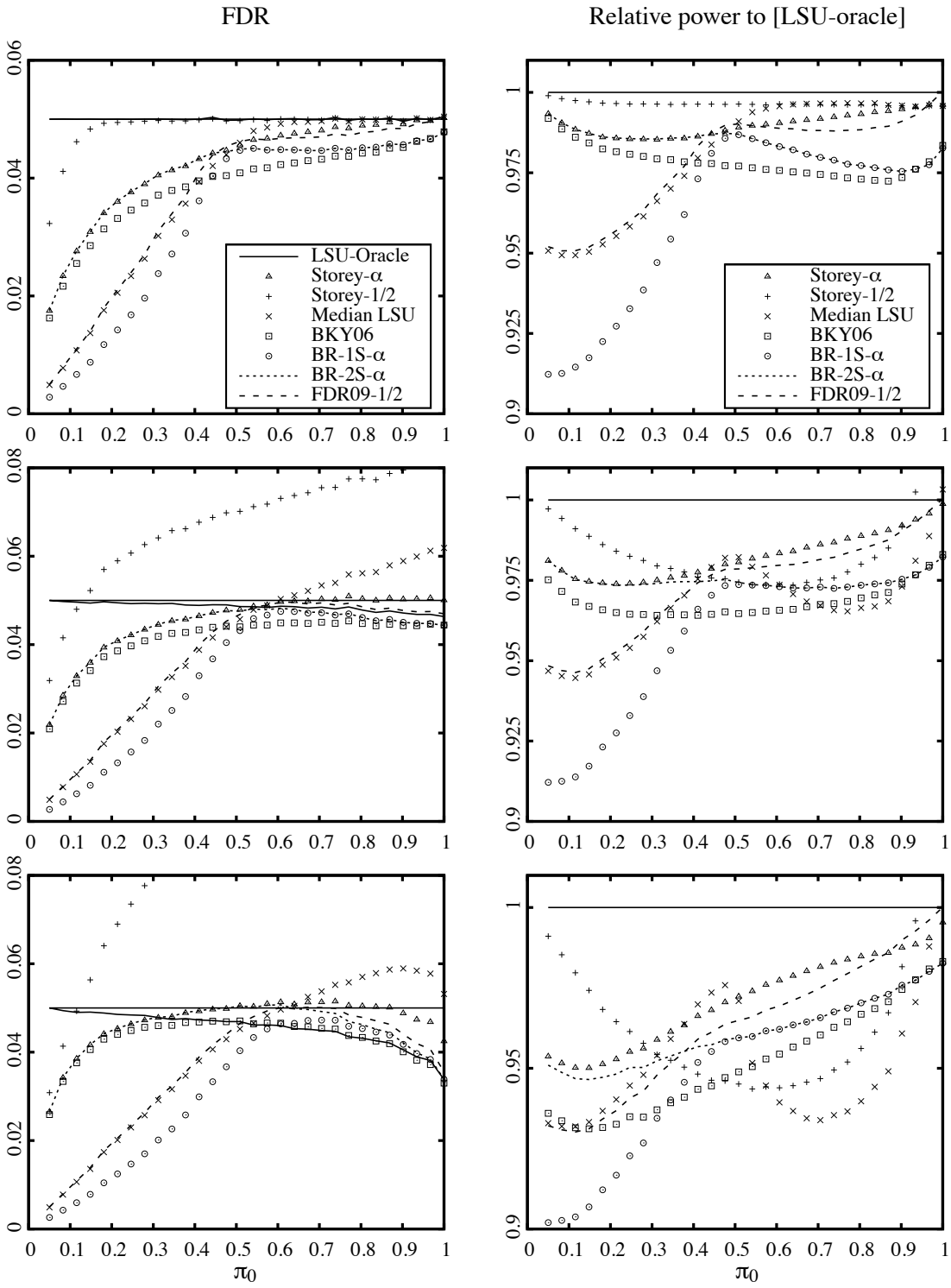


Figure 2: FDR and power relative to oracle as a function of the true proportion π_0 of null hypotheses . Target FDR is $\alpha = 5\%$, total number of hypotheses $m = 100$. The mean for the alternatives is $\bar{\mu} = 3$. From top to bottom: pairwise correlation coefficient $\rho \in \{0, 0.2, 0.5\}$.

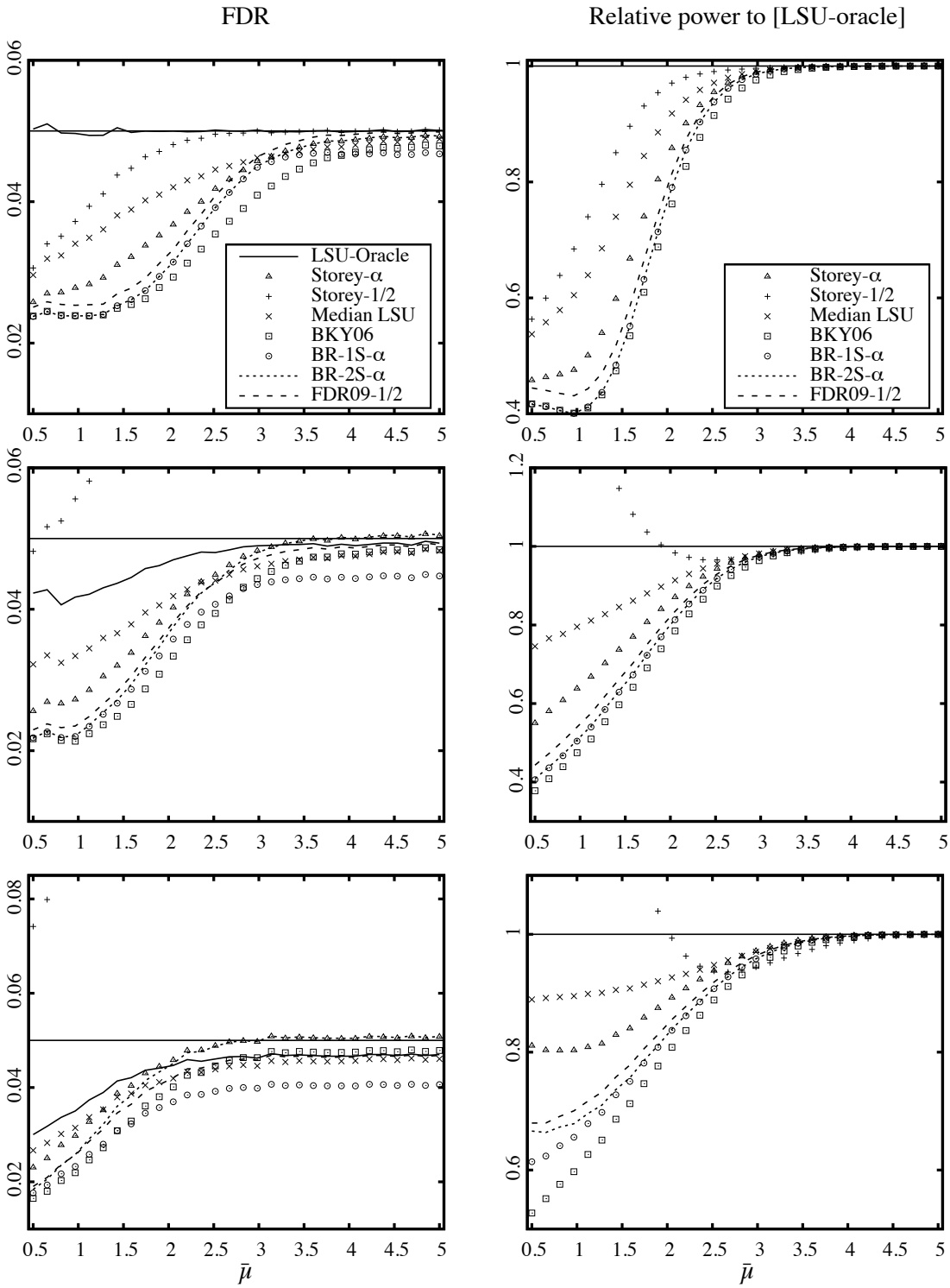


Figure 3: FDR and power relative to oracle as a function of the common alternative hypothesis mean $\bar{\mu}$. Target FDR is $\alpha = 5\%$, total number of hypotheses $m = 100$. The proportion of true null hypotheses is $\pi_0 = 0.5$. From top to bottom: pairwise correlation coefficient $\rho \in \{0, 0.2, 0.5\}$.

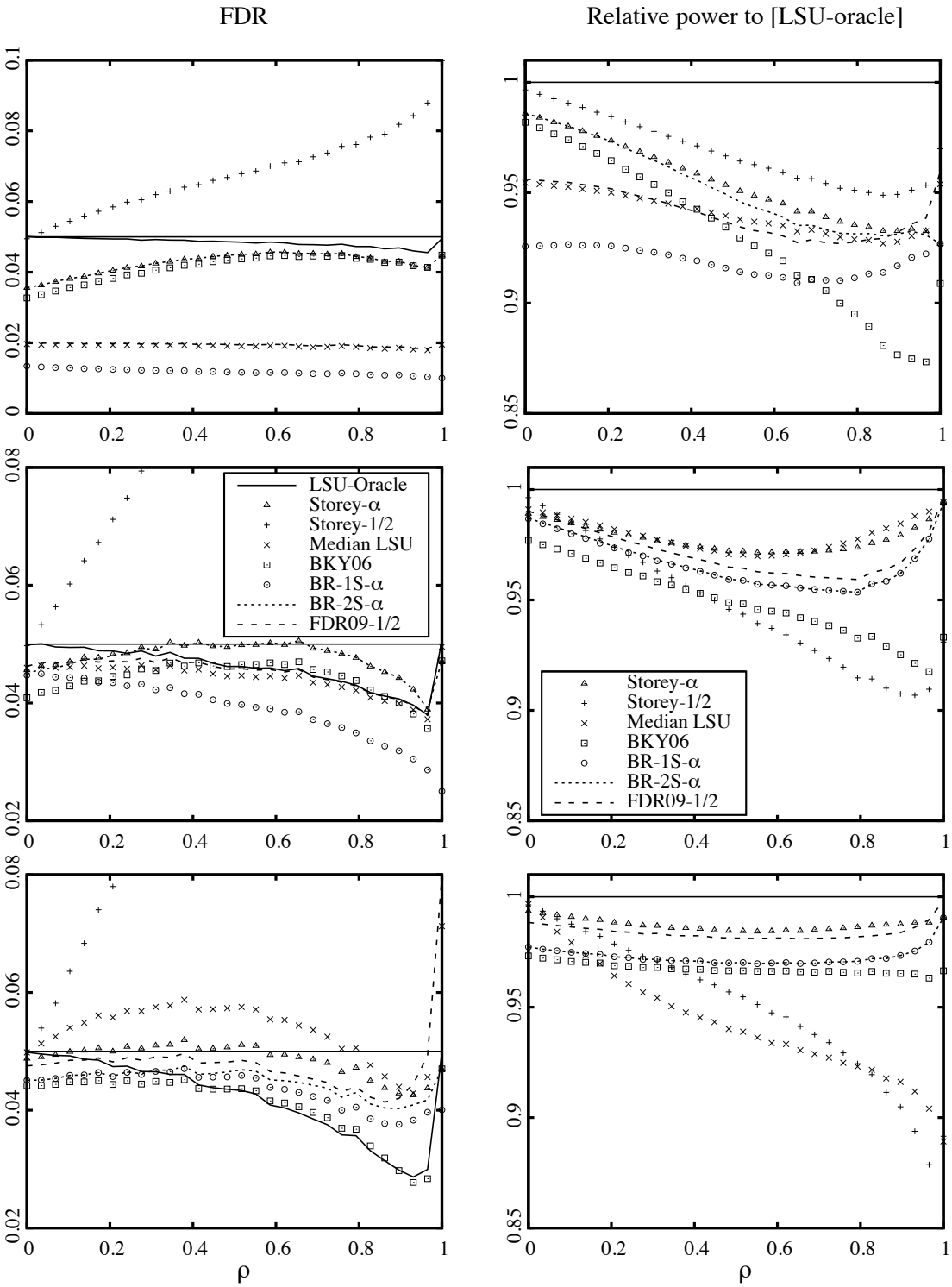


Figure 4: FDR and power relative to oracle as a function of the pairwise correlation coefficient ρ .Target FDR is $\alpha = 5\%$, total number of hypotheses $m = 100$. The mean for the alternatives is $\bar{\mu} = 3$. From top to bottom: proportion of true null hypotheses $\pi_0 \in \{0.2, 0.5, 0.8\}$.

The other remaining procedures seem to exhibit a robust control of the FDR under dependence, or at least their FDR appears to be very close to the target level (except for [FDR09-$\frac{1}{2}$] when ρ and π_0 are close to 1). For these procedures, it seems that the qualitative conclusions concerning power comparison found in the independent case remain true. To sum up:

- the best overall procedure seems to be [Storey-α]: its FDR seems to be under or only slightly over the target level in all situations, and it exhibits globally a power superior to other procedures.
- then come in order of power, our two-stage procedure [BR08-2S-α], then [BKY06-α].
- like in the dependent case, [FDR09-$\frac{1}{2}$] ranks second when $\pi_0 > \frac{1}{2}$ but tends to perform noticeably poorer if π_0 gets smaller. Its FDR is also not controlled if very strong correlations are present.

The overall conclusion we draw from these experiments is that for practical use, we recommend in priority [Storey-α], then as close seconds [BR08-2S-α] or [FDR09-$\frac{1}{2}$] (the latter when it is expected that $\pi_0 > 1/2$, and that there are no very strong correlations present). The procedudre [BKY06-α] is also competitive but appears to be in most cases noticeably outperformed by the above ones. These procedures all exhibit good robustness to dependence for FDR control as well as comparatively good power. The fact that [Storey-α] performs so well and seems to hold the favorite position has up to our knowledge not been reported before (it was not included in the simulations of Benjamini et al., 2006) and came somewhat as a surprise to us.

Remark 18 *As pointed out earlier, the fact that [FDR09-$\frac{1}{2}$] performs sub-optimally for $\pi_0 < \frac{1}{2}$ appears to be strongly linked to the choice of parameter $\eta = \frac{1}{2}$. Namely, the implicit estimator of π_0^{-1} in the procedure is capped at η (see Remark 15). Choosing a higher value for η will reduce the sub-optimality region but increase the variability of the estimate and thus decrease the overall robustness of the procedure (if dependence is present; and also under independence if only a small number m of hypotheses are tested, as for this procedure the convergence of the FDR towards its asymptotically controlled value becomes slower as η grows towards 1).*

Remark 19 *Another two-stage adaptive procedure was introduced in Sarkar (2008a), which is very similar to a plug-in procedure using [Storey-λ]. In fact, in the experiments presented in Sarkar (2008a), the two procedures are almost equivalent, corresponding to $\lambda = 0.995$. We decided not to include this additional procedure in our simulations to avoid overloading the plots. Qualitatively, we observed that the procedures of Sarkar (2008a) or [Storey-0.995] are very similar in behavior to [Storey-$\frac{1}{2}$]: very performant in the independent case but very fragile with respect to deviations from independence.*

Remark 20 *One could formulate the concern that the observed FDR control for [Storey-α] could possibly fail with other parameters settings, for example when π_0 and/or ρ are close to one. We performed additional simulations to this respect (a more detailed report is available on the authors' web pages), which we summarize briefly here. We considered the following cases: $\pi_0 = 0.95$ and varying $\rho \in [0,1]$; $\rho = 0.95$ and varying $\pi_0 \in [0,1]$; finally (π_0, ρ) varying both in $[0.8,1]^2$, using a finer discretization grid to cover this region in more detail. In all the above cases Storey-α still had its FDR very close to (or below) α. Note also that the case $\rho \simeq 1$ and $\pi_0 \simeq 1$ is in accordance with*

the result of Section 3.3, stating that $\mathrm{FDR}(\textit{Storey-}\alpha) = \alpha$ *when* $\rho = 1$ *and* $\pi_0 = 1$. *Finally, we also performed additional experiments for different choices of the number of hypotheses to test (*$m = 20$ *and* $m = 10^4$*) and different choices of the target level (*$\alpha = 10\%, 1\%$*). In all of these cases were the results qualitatively in accordance with the ones already presented here.*

4. New Adaptive Procedures with Provable FDR Control under Arbitrary Dependence

In this section, we consider from a theoretical point of view the problem of constructing multiple testing procedures that are adaptive to π_0 under arbitrary dependence conditions of the *p*-values. The derivation of adaptive procedures that have provably controlled FDR under dependence appears to have been only studied scarcely (see Sarkar, 2008a, and Farcomeni, 2007). Here, we propose to use a two-stage procedure where the first stage is a multiple testing with either controlled FWER or controlled FDR. The first option is relatively straightfoward and is intended as a reference. In the second case, we use Markov's inequality to estimate π_0^{-1}. Since Markov's inequality is general but not extremely precise, the resulting procedures are obviously quite conservative and are arguably of a limited practical interest. However, we will show that they still provide an improvement, in a certain regime, with respect to the (non-adaptive) LSU procedure in the PRDS case and with respect to the family of (non-adaptive) procedures proposed in Theorem 7 in the arbitrary dependence case.

For the purposes of this section, we first recall the formal definition for PRDS dependence of Benjamini and Yekutieli (2001):

Definition 21 (PRDS condition) *Remember that a set* $D \subset [0,1]^{\mathcal{H}}$ *is said to be nondecreasing if for all* $x, y \in [0,1]^{\mathcal{H}}$*, if* $x \leq y$ *coordinate-wise,* $x \in D$ *implies* $y \in D$*. Then, the p-value family* $\mathbf{p} = (p_h, h \in \mathcal{H})$ *is said to be positively regression dependent on each one from* $\mathcal{H}_0$ *(PRDS on* $\mathcal{H}_0$ *in short) if for any nondecreasing measurable set* $D \subset [0,1]^{\mathcal{H}}$ *and for all* $h \in \mathcal{H}_0$*, the function* $u \in [0,1] \mapsto \mathbb{P}[\mathbf{p} \in D \mid p_h = u]$ *is nondecreasing.*

On the one hand, it was proved by Benjamini and Yekutieli (2001) that the LSU still has controlled FDR at level $\pi_0\alpha$ (i.e., Theorem 6 still holds) under the PRDS assumption. On the other hand, under totally arbitrary dependence this result does not hold, and Theorem 7 provides a family of threshold collection resulting in controlled FDR at the same level in this case.

Our first result concerns a two-stage procedure where the first stage R_0 is any multiple testing procedure with controlled FWER, and where we (over-) estimate m_0 via the straightforward estimator $(m - |R_0|)$. This should be considered as a form of baseline reference for this type of two-stage procedure.

Theorem 22 *Let* R_0 *be a nonincreasing multiple testing procedure and assume that its FWER is controlled at level* α_0 *, that is,* $\mathbb{P}[R_0 \cap \mathcal{H}_0 \neq \emptyset] \leq \alpha_0$ *. Then the adaptive step-up procedure R with data-dependent threshold collection* $\Delta(i) = \alpha_1 (m - |R_0|)^{-1} \beta(i)$ *has FDR controlled at level* $\alpha_0 + \alpha_1$ *in either of the following dependence situations:*

- *the p-value family* $(p_h, h \in \mathcal{H})$ *is PRDS on* $\mathcal{H}_0$ *and the shape function is the identity function.*
- *the p-values have unspecified dependence and* β *is a shape function of the form (3).*

Here it is clear that the price for adaptivity is a certain loss in FDR control for being able to use the information of the first stage. If we choose $\alpha_0 = \alpha_1 = \alpha/2$, then this procedure will outperform its

non-adaptive counterpart (using the same shape function) only if there are more than 50% rejected hypotheses in the first stage. Only if it is expected that this situation will occur does it make sense to employ this procedure, since it will otherwise perform worse than the non-adaptive procedure.

Our second result is a two-stage procedure where the first stage has controlled FDR. First introduce, for a fixed constant $\kappa \geq 2$, the following function: for $x \in [0,1]$,

$$F_\kappa(x) = \begin{cases} 1 & \text{if } x \leq \kappa^{-1} \\ \frac{2\kappa^{-1}}{1-\sqrt{1-4(1-x)\kappa^{-1}}} & \text{otherwise} \end{cases}.$$

If R_0 denotes the first stage, we propose using $F_\kappa(|R_0|/m)$ as an (under-)estimation of π_0^{-1} at the second stage. We obtain the following result:

Theorem 23 *Let β be a fixed shape function, and $\alpha_0, \alpha_1 \in (0,1)$ such that $\alpha_0 \leq \alpha_1$. Denote by R_0 the step-up procedure with threshold collection $\Delta_0(i) = \alpha_0\beta(i)/m$. Then the adaptive step-up procedure R with data-dependent threshold collection $\Delta_1(i) = \alpha_1\beta(i)F_\kappa(|R_0|/m)/m$ has FDR upper bounded by $\alpha_1 + \kappa\alpha_0$ in either of the following dependence situations:*

- *the p-value family $(p_h, h \in \mathcal{H})$ is PRDS on $\mathcal{H}_0$ and the shape function is the identity function.*
- *the p-values have unspecified dependence and β is a shape function of the form (3).*

For instance, in the PRDS case, the procedure R of Theorem 23, used with $\kappa = 2$, $\alpha_0 = \alpha/4$ and $\alpha_1 = \alpha/2$, corresponds to the adaptive linear step-up procedure at level $\alpha/2$ with the following estimator for π_0^{-1} :

$$\frac{1}{1-\sqrt{(2|R_0|/m-1)_+}},$$

where $|R_0|$ is the number of rejections of the LSU procedure at level $\alpha/4$.

Whether in the PRDS or arbitrary dependence case, with the above choice of parameters, we note that R is less conservative than the non-adaptive step-up procedure with threshold collection $\Delta(i) = \alpha\beta(i)/m$ if $F_2(|R_0|/m) \geq 2$ or equivalently when R_0 rejects more than $F_2^{-1}(2) = 62,5\%$ of the null hypotheses. Conversely, R is more conservative otherwise, and we can lose up to a factor 2 in the threshold collection with respect to the standard one-stage version. Therefore, here again this adaptive procedure is only useful in the cases where it is expected that a "large" proportion of null hypotheses can easily be rejected. In particular, when we use Theorem 23 under unspecified dependence, it is relevant to choose the shape function β from a distribution ν concentrated on the large numbers of $\{1,\ldots,m\}$. Finally, note that it is not immediate to see if this procedure will improve on the one of Theorem 22. Namely, with the above choice of parameters, procedure of Theorem 22 has the advantage of using a better estimator of π_0^{-1} of the form $(1-x)^{-1} \geq (1-\sqrt{(2x-1)_+})^{-1}$ in the second round (with $x = |R_0|/m$ coming from the first round), but it has the drawback to use a first round controlling the FWER at level $\alpha/2$ which can be much more conservative than controlling the FDR at level $\alpha/4$.

To explore this issue, we performed the two above procedures, in a favorable situation where π_0 is small. Namely, we considered the simulation setting of Section 3.4 with $\rho = 0.1$, $m_0 = 100$ and $m = 1000$ (hence $\pi_0 = 10\%$) and $\alpha = 5\%$. The common value $\bar{\mu}$ of the positive means varies in the range $[0,5]$. Larger values of $\bar{\mu}$ correspond to a very large proportion of hypotheses that are easy to

reject, which favors the first stage of the two above procedures. A positively correlated family of Gaussians satisfies the PRDS assumption (see Benjamini and Yekutieli, 2001), so that we use the identity shape function (linear step-up), and compare our procedures against the standard LSU. For the FWER-controlled first stage of Theorem 22, we chose a standard Holm procedure (see Holm, 1979), which is a step-down procedure with threshold collection $t(i) = \alpha m/(m-i+1)$. In Figure 5, we report the relative power to the oracle LSU, and the False Non-discovery Rate (FNR), which is the converse of the FDR for type II errors, that is, the average of the ratio of non-rejected false hypotheses over the total number of non-rejected hypotheses. Since we are in a situation where π_0 is small, the FNR might actually be a more relevant criterion than the raw power: in this situation, because of the small number of non-rejected hypotheses, two different procedures could have their power very similar and close to 1, but noticeably different FNRs.

The conclusion is that there exists an (unfortunately relatively small) region where the adaptive procedures improve over the standard LSU in terms of power. In terms of FNR, the improvement is more noticeable and over a larger region. Finally, our two-stage adaptive procedure of Theorem 23 appears to outperform consistently the baseline of Theorem 22. These results are still unsatisfying to the extent that the adaptive procedure improves over the non-adaptive one only in a region limited to some quite particular cases, and underperforms otherwise. Nevertheless, this demonstrates theoretically the possibility of provably adaptive procedures under dependence. Again, this theme appears to have been theoretically studied in only a handful of previous works until now, and improving significantly the theory in this setting is still an open challenge.

Remark 24 *Some theoretical results for two-stage procedures under possible dependence using a first stage with controlled FWER or controlled FDR appeared earlier (Farcomeni, 2007). However, it appears that in this reference, it is implicitly assumed that the two stages are actually independent, because the proof relies on a conditioning argument wherein FDR control for the second stage still holds conditionally on the first stage output. This is the case for example if the two stages are performed on separate families of p-values corresponding to a new independent observation. Here we specifically wanted to take into account that we use the same collection of p-values for the two stages, and therefore that the two stages cannot assumed to be independent. In this sense the result of Theorem 22 is novel with respect to that of Farcomeni (2007).*

Remark 25 *The theoretical problem of adaptive procedures under arbitrary dependence was also considered by Sarkar (2008a) using two-stage procedures. However, the procedures proposed there were reported not to yield any significant improvement over non-adaptive procedures.*

5. Conclusion and Discussion

We proposed several adaptive multiple testing procedures that provably control the FDR under different hypotheses on the dependence of the p-values. Firstly, we introduced the one- and two-stage procedures *BR-1S* and *BR-2S* and we proved their theoretical validity when the p-values are independent. The procedure *BR-2S* is less conservative in general (except in marginal situations) than the adaptive procedure proposed by Benjamini et al. (2006). Extensive simulations showed that these new procedures appear to be robustly controlling the FDR even in a positive dependence situation, which is a very desirable property in practice. This is an advantage with respect to the [Storey-$\frac{1}{2}$] procedure, which is less conservative but breaks down under positive dependence. Moreover, our simulations showed that the choice of parameter $\lambda = \alpha$ instead of $\lambda = 1/2$ in the Storey procedure

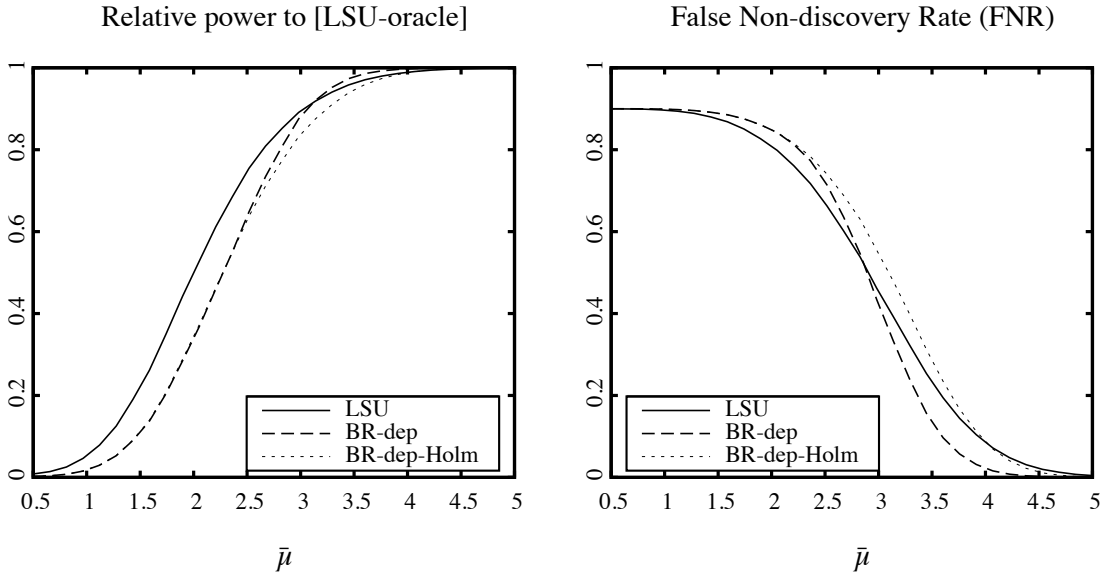


Figure 5: Relative power to oracle and false non-discovery rate (FNR) of the different procedures, as a function of the common alternative hypothesis mean $\bar{\mu}$. Parameters are $\alpha = 5\%$, $m = 1000$, $\pi_0 = 10\%$, $\rho = 0.1$. "BR08-dep-Holm" corresponds to the procedure of Theorem 22 using $\alpha_1 = \alpha_0 = \alpha/2$ and Holm's step-down for the first step, and "BR08-dep" to the procedure of Theorem 23 with $\kappa = 2$, $\alpha_0 = \alpha/4$ and $\alpha_1 = \alpha/2$. The shape function β is the identity function. Each point is estimated by an average over 10^4 independent repetitions.

resulted in a much more robust procedure under positive dependence, at the price of being slightly more conservative. This fact is supported by a theoretical investigation of the maximally dependent case. These properties do not appear to have been reported before, and put forward Storey-α as a procedure of considerable practical interest.

Secondly, we presented what we think is among the first examples of adaptive multiple testing procedures with provable FDR control in the PRDS case and under unspecified dependence. An important difference with respect to earlier works on this topic is that the procedures we introduced here are both theoretically founded and can be shown to improve over non-adaptive procedures in certain (admittedly limited) circumstances. Although their interest at this point is mainly theoretical, this shows in principle that adaptivity can improve performance in a theoretically rigorous way even without the independence assumption.

The proofs of the results have been built upon the notion of *self-consistency* and other technical tools introduced in a previous work (Blanchard and Roquain, 2008). We believe these tools allow for a more unified approach than in the classical adaptive multiple testing literature, avoiding in particular to deal explicitly with the reordered p-values, which can be somewhat cumbersome.

Another advantage of this approach is that it can be extended in a relatively straightforward manner to the case of *weighted FDR*, that is, the quantity (2) where the cardinality measure $|.|$ has been replaced by a general measure $W(R) = \sum_{h \in R} w_h$ (with $W(\mathcal{H}) = \sum_{h \in \mathcal{H}} w_h = m$). This allows

in particular to recover results very similar to those of Benjamini and Heller (2007) and can also be used to prove that a (generalized) Storey estimator can be used to control the weighted FDR. The modifications needed to include this generalizations are relatively minor; we omit the details here and refer the reader to Blanchard and Roquain (2008) to see how the case of weighted FDR can be handled using the same technical tools.

There remains a vast number of open issues concerning adaptive procedures. We first want to underline once more that the theory for adaptive procedures under dependence is still underdeveloped. It might actually be too restrictive to look for procedures having theoretically controlled FDR uniformly over arbitrary dependence situations such as what we studied in Section 4. An interesting future theoretical direction could be to prove that some of the adaptive procedures showing good robustness in our simulations actually have controlled FDR under some types of dependence, at least when the p-values are in some sense not too far from being independent.

6. Proofs

This section collects proofs for all the stated results, following their order of appearance in the text.

6.1 Proofs for Section 3

The following proofs use the notation $\mathbf{p}_{0,h}$ and $\mathbf{p}_{-h}$ defined at the beginning of Section 3.2.

6.1.1 PROOF OF THEOREM 9

Let R denote a nonincreasing self-consistent procedure with respect to Δ defined in (4). By definition, R satisfies

$$R \subset \left\{ h \in \mathcal{H} \mid p_h \leq \min\left((1-\lambda)\frac{\alpha|R|}{m-|R|+1}, \lambda \right) \right\}.$$

Therefore, we have

$$\begin{aligned}
\mathrm{FDR}(R) &= \sum_{h\in\mathcal{H}_0} \mathbb{E}\left[\frac{\mathbf{1}\{h \in R(\mathbf{p})\}}{|R(\mathbf{p})|} \right] \\
&\leq \sum_{h\in\mathcal{H}_0} \mathbb{E}\left[\frac{\mathbf{1}\left\{ p_h \leq (1-\lambda)\frac{\alpha|R(\mathbf{p})|}{m-|R(\mathbf{p})|+1} \right\}}{|R(\mathbf{p})|} \right] \\
&\leq \sum_{h\in\mathcal{H}_0} \mathbb{E}\left[\frac{\mathbf{1}\left\{ p_h \leq (1-\lambda)\frac{\alpha|R(\mathbf{p})|}{m-|R(\mathbf{p}_{0,h})|+1} \right\}}{|R(\mathbf{p})|} \right] \\
&= \sum_{h\in\mathcal{H}_0} \mathbb{E}\left[\mathbb{E}\left[\frac{\mathbf{1}\left\{ p_h \leq (1-\lambda)\frac{\alpha|R(\mathbf{p})|}{m-|R(\mathbf{p}_{0,h})|+1} \right\}}{|R(\mathbf{p})|} \middle| \mathbf{p}_{-h} \right] \right] \\
&\leq (1-\lambda)\alpha \sum_{h\in\mathcal{H}_0} \mathbb{E}\left[\frac{1}{m-|R(\mathbf{p}_{0,h})|+1} \right],
\end{aligned}$$

The second inequality above comes from $|R(\mathbf{p})| \leq |R(\mathbf{p}_{0,h})|$, which itself holds because $|R|$ is coordinate-wise nonincreasing in each p-value. The last inequality is obtained with Lemma 27

of Section 7 with $U = p_h$, $g(U) = |R(\mathbf{p}_{-h}, U)|$ and $c = \frac{(1-\lambda)\alpha}{m-|R(\mathbf{p}_{0,h})|+1}$, because the distribution of p_h conditionally on $\mathbf{p}_{-h}$ is (by independence) identical to its marginal distribution, hence stochastically lower bounded by a uniform variable on $[0,1]$; $|R|$ is coordinate-wise nonincreasing; and because $\mathbf{p}_{0,h}$ depends only on the p-values of $\mathbf{p}_{-h}$. Finally, since the threshold collection of R is upper bounded by λ, we get

$$(1-\lambda)\mathbb{E}\left[m/(m-|R(\mathbf{p}_{0,h})|+1)\right] \leq \mathbb{E}G_1(\mathbf{p}_{0,h}),$$

where G_1 is the Storey estimator with parameter λ. We then use $\mathbb{E}G_1(\mathbf{p}_{0,h}) \leq \pi_0^{-1}$ (see proof of Corollary 13) to conclude. ■

6.1.2 PROOF OF LEMMA 10

Denote $G(t) = \pi_0 t + (1-\pi_0)F(t)$ the c.d.f. of the p-values under the random effects mixture model. Let us denote by $\hat{t}_m$ the threshold of the LSU procedure. The proportion of rejected hypotheses from the initial pool is then exactly $\widehat{G}_m(\hat{t}_m)$, where $\widehat{G}_m$ is the empirical cdf of the p-values. It was proved by Genovese and Wasserman (2002) under the random effects model, that as m tends to infinity the LSU threshold $\hat{t}_m$ converges in probability to $t^\star$, which is the largest point $t \in [0,1]$ such that $G(t) = \alpha^{-1}t$. Since $\widehat{G}_m$ converges in probability uniformly to G , we deduce that the proportion of rejected hypotheses converges to $\alpha^{-1}t^*$ in probability; hence, if $t^* > \alpha^2$, the probability that the proportion of rejected hypotheses is less that $\alpha + 1/m$ converges to zero; and conversely converges to 1 if $t^* < \alpha^2$.

The definition of t^* and the expression for G in the Gaussian mean shift model imply the following relation whenever $t^* > 0$:

$$\mu = \overline{\Phi}^{-1}(t^\star) - \overline{\Phi}^{-1}\left(\frac{\alpha^{-1}-\pi_0}{1-\pi_0}t^\star\right).$$

It is easily seen that if $\pi_0 < (1+\alpha)^{-1}$, the quantity μ^* in the statement of the lemma is well defined and we have $t^* > \alpha^2$ for $\mu > \mu^*$. This gives the first part of the result.

Conversely, if $\pi_0 > (1+\alpha)^{-1}$ we have $t^* = 0$, and if $\pi_0 < (1+\alpha)^{-1}$ but $\mu < \mu^*$, we have $t^* < \alpha^2$; this leads to the second part of the result. ■

6.1.3 PROOF OF THEOREM 11

By definition of self-consistency, the procedure R satisfies

$$R \subset \{h \in \mathcal{H} \mid p_h \leq \alpha |R| G(\mathbf{p})/m\}.$$

Therefore,

$$\mathrm{FDR}(R) = \sum_{h\in\mathcal{H}_0} \mathbb{E}\left[\frac{\mathbf{1}\{h \in R(\mathbf{p})\}}{|R(\mathbf{p})|}\right] \leq \sum_{h\in\mathcal{H}_0} \mathbb{E}\left[\frac{\mathbf{1}\{p_h \leq \alpha |R(\mathbf{p})| G(\mathbf{p})/m\}}{|R(\mathbf{p})|}\right].$$

Since G is nonincreasing, we get:

$$\begin{aligned}
\mathrm{FDR}(R) &\leq \sum_{h\in\mathcal{H}_0} \mathbb{E}\left[\frac{\mathbf{1}\{p_h \leq \alpha |R(\mathbf{p})| G(\mathbf{p}_{0,h})/m\}}{|R(\mathbf{p})|}\right] \\
&= \sum_{h\in\mathcal{H}_0} \mathbb{E}\left[\mathbb{E}\left[\frac{\mathbf{1}\{p_h \leq \alpha |R(\mathbf{p})| G(\mathbf{p}_{0,h})/m\}}{|R(\mathbf{p})|}\middle| \mathbf{p}_{-h}\right]\right] \leq \frac{\alpha}{m}\sum_{h\in\mathcal{H}_0} \mathbb{E}G(\mathbf{p}_{0,h}).
\end{aligned}$$

The last step is obtained with Lemma 27 of Section 7 with $U = p_h$, $g(U) = |R(\mathbf{p}_{-h}, U)|$ and $c = \alpha G(\mathbf{p}_{0,h})/m$, because the distribution of p_h conditionally on $\mathbf{p}_{-h}$ is (by independence) identical to its marginal distribution, hence stochastically lower bounded by a uniform variable; $|R|$ is coordinate-wise nonincreasing; and $\mathbf{p}_{0,h}$ depends only on the p-values of $\mathbf{p}_{-h}$. ■

6.1.4 PROOF OF COROLLARY 12

Assuming $\mathcal{H}_0 \neq \emptyset$ without loss of generality, for $h_0 \in \mathcal{H}_0$ we want to upper bound $\mathbb{E}[G(\mathbf{p}_{0,h_0})]$ appearing in the bound of Theorem 11. Let $\widetilde{\mathbf{p}}_{0,h_0}$ denote the family of p-values $\mathbf{p}_{0,h_0}$ where all p-values p_h , $h \in \mathcal{H} \setminus \mathcal{H}_0$ have been replaced by zero. Since G is nonincreasing, we have $\mathbb{E}[G(\mathbf{p}_{0,h_0})] \leq \mathbb{E}[G(\widetilde{\mathbf{p}}_{0,h_0})]$. Now, for any $h \in \mathcal{H}_0 \setminus \{h_0\}$, denote $\widetilde{\mathbf{p}}'_{0,h_0}$ the family $\widetilde{\mathbf{p}}_{0,h_0}$, where the variable p_h has been replaced by u_h , an independent uniform variable on $[0,1]$. Since both p_h and u_h are independent of the other p-values, p_h is stochastically lower bounded by u_h and G is nonincreasing, we have

$$\mathbb{E}\left[G(\widetilde{\mathbf{p}}_{0,h_0})\middle|\mathbf{p}_{-h}\right] \leq \mathbb{E}\left[G(\widetilde{\mathbf{p}}'_{0,h_0})\middle|\mathbf{p}_{-h}\right] ,$$

hence also in unconditional expectation. Iterating this reasoning in succession for all $h \in \mathcal{H}_0 \setminus \{h_0\}$, we have finally replaced $\mathbf{p}_{0,h_0}$ by a family of $m_0 - 1$ independent uniform variables and $m - m_0 + 1$ zeros, while only increasing the expected value, so that (now using that G is permutation invariant)

$$\mathbb{E}[G(\mathbf{p}_{0,h_0})] \leq \mathbb{E}_{\mathbf{p}\sim DU(m,m_0-1)}[G(\mathbf{p})] ,$$

which, combined with (6), entails the desired result. ■

6.1.5 PROOF OF COROLLARY 13

First, we prove that the sufficient condition of Theorem 11 holds for the nonincreasing estimators G_i, $i = 1,3,4$. To that end, we reproduce here without major changes the arguments used by Benjamini et al. (2006). The bound for G_1 is obtained using Lemma 30 (see below) with $k = m_0$ and $q = 1-\lambda$: for all $h \in \mathcal{H}_0$,

$$\mathbb{E}[G_1(\mathbf{p}_{0,h})] \leq m(1-\lambda)\mathbb{E}\left[\left(\sum_{h' \in \mathcal{H}_0 \setminus \{h\}} \mathbf{1}\{p_{h'} > \lambda\} + 1\right)^{-1}\right] \leq \pi_0^{-1}.$$

The proof for G_3 and G_4 is deduced from the one of G_1 because G_3 and G_4 are smaller than G_1 pointwise.

Secondly, for G_2 we use a somewhat more direct argument than Benjamini et al. (2006), namely using Corollary 12 and proving that $\gamma(G_2, m) \leq 1$. Take $\mathbf{p} \sim DU(m, m_0 - 1)$. On the one hand, if $k_0 \leq m - m_0 + 1$, we have $p_{(k_0)} = 0$, and therefore $\pi_0 G_2(\mathbf{p}) = \pi_0 m/(m - k_0 + 1) \leq 1$ pointwise. On the other hand, if $k_0 \geq m - m_0 + 2$, we have $p_{(k_0)} = q_{(k_0 - m + m_0 - 1)}$, where $q_{(1)} \leq \ldots \leq q_{(m_0-1)}$ are the $(m_0 - 1)$ ordered p-values of $\mathbf{p}$ corresponding to uniform variables. Thus,

$$\pi_0 \mathbb{E}[G_2(\mathbf{p})] = m\pi_0 \frac{1 - \mathbb{E}\left[q_{(k_0-m+m_0-1)}\right]}{m - k_0 + 1} = m\pi_0 \frac{1 - (k_0 - m + m_0 - 1)/m_0}{m - k_0 + 1} = 1.$$

6.1.6 PROOF OF PROPOSITION 17

Let us first consider adaptive one-stage procedures: for any step-up procedure R of threshold $\Delta(i) = \alpha\beta(i)/m$ we easily derive that the probability that R makes any rejection is

$$\mathbb{P}[\exists i \mid p_i \leq \Delta(i)] = \mathbb{P}[\exists i \mid p_1 \leq \Delta(i)] = \mathbb{P}[p_1 \leq \Delta(m)] = \Delta(m),$$

which is FDR(R) because $m_0 = m$. The results for *BR-1S*-λ and *FDR09*-η follow.

With the same reasoning, we find that for any plug-in adaptive linear step-up procedure R that uses an estimator $G(\mathbf{p})$,

$$\mathrm{FDR}(R) = \mathbb{P}\left[p_1 \leq \alpha G(\mathbf{p})\right]. \tag{7}$$

Next, for the Storey plug-in procedure, we have $G_1(p_1,...,p_1) = (1-\lambda)m/(m\mathbf{1}\{p_1 > \lambda\}+1)$, so that applying (7), we get

$$\begin{aligned}\mathrm{FDR}(\textit{Storey-}\lambda) &= \mathbb{P}\left[p_1 \leq \alpha G_1(\mathbf{p})\right] \\ &= \mathbb{P}\left[p_1 \leq \lambda,\, p_1 \leq \alpha(1-\lambda)m\right] + \mathbb{P}\left[p_1 > \lambda,\, p_1 \leq \alpha(1-\lambda)m/(m+1)\right] \\ &= \min\left(\lambda, \alpha(1-\lambda)m\right) + \left(\frac{\alpha(1-\lambda)m}{m+1} - \lambda\right)_+ .\end{aligned}$$

For the quantile procedure, we have

$$\mathbb{P}\left[p_1 \leq \alpha(1-p_1)m/(m-k_0+1)\right] = \mathbb{P}\left[p_1((1+\alpha)m-k_0+1) \leq \alpha m\right] = \frac{\alpha}{1+\alpha-(k_0-1)/m}.$$

For the BKY06 procedure, we simply remark that since the linear step-up procedure of level λ rejects all the hypotheses when $p_1 \leq \lambda$ and rejects no hypothesis otherwise, the estimator G_1 and G_3 are equal in this case. The proof for *BR-2S*-λ is similar. ■

6.2 Proofs for Section 4

We begin with a technical lemma that will be useful for proving both Theorem 22 and 23. It is related to techniques previously introduced by Blanchard and Roquain (2008).

Lemma 26 *Assume R is a multiple testing procedure satisfying the self-consistency condition:*

$$R \subset \left\{h \in \mathcal{H} \mid p_h \leq \alpha G(\mathbf{p})\beta(|R|)/m\right\},$$

where $G(\mathbf{p})$ is a data-dependent factor. Then the following inequality holds:

$$FDR(R) \leq \alpha + \mathbb{E}\left[\frac{|R\cap\mathcal{H}_0|}{|R|}\mathbf{1}\{|R|>0\}\mathbf{1}\left\{G(\mathbf{p}) > \pi_0^{-1}\right\}\right], \tag{8}$$

under either of the following conditions:

- *the p-value family $(p_h, h \in \mathcal{H})$ is PRDS on $\mathcal{H}_0$, R is nonincreasing and β is the identity function.*
- *the p-values have unspecified dependence and β is a shape function of the form (3).*

Proof. We have

$$\begin{aligned}\mathrm{FDR}(R) &= \mathbb{E}\left[\frac{|R\cap\mathcal{H}_0|}{|R|}\mathbf{1}\{|R|>0\}\right] \\ &= \mathbb{E}\left[\frac{|R\cap\mathcal{H}_0|}{|R|}\mathbf{1}\{|R|>0\}\mathbf{1}\left\{G \leq \pi_0^{-1}\right\}\right] + \mathbb{E}\left[\frac{|R\cap\mathcal{H}_0|}{|R|}\mathbf{1}\{|R|>0\}\mathbf{1}\left\{G > \pi_0^{-1}\right\}\right] \\ &\leq \sum_{h\in\mathcal{H}_0}\mathbb{E}\left[\frac{\mathbf{1}\{p_h \leq \alpha\beta(|R|)/m_0\}}{|R|}\right] + \mathbb{E}\left[\frac{|R\cap\mathcal{H}_0|}{|R|}\mathbf{1}\{|R|>0\}\mathbf{1}\left\{G > \pi_0^{-1}\right\}\right].\end{aligned}$$

The desired conclusion will therefore hold if we establish that for any $h \in \mathcal{H}_0$, and $c > 0$:

$$\mathbb{E}\left[\frac{\mathbf{1}\{p_h \leq c\beta(|R|)\}}{|R|}\right] \leq c\,.$$

Under unspecified dependence, this is a direct consequence of Lemma 29 of Section 7 with $U = p_h$ and $V = \beta(|R|)$. For the PRDS case, we note that since $|R(\mathbf{p})|$ is coordinate-wise nonincreasing in each p-value, for any $v > 0$, $D = \{\mathbf{z} \in [0,1]^{\mathcal{H}} \mid |R(\mathbf{z})| < v\}$ is a measurable nondecreasing set, so that the PRDS property implies that $u \mapsto \mathbb{P}[|R| < v \mid p_h = u]$ is nondecreasing. This implies that $u \mapsto \mathbb{P}[|R| < v \mid p_h \leq u]$ by the following argument (see also Lehmann, 1966, cited by Benjamini and Yekutieli, 2001, and Blanchard and Roquain, 2008): putting $\gamma = \mathbb{P}[p_h \leq u \mid p_h \leq u']$,

$$\begin{aligned}\mathbb{P}\left[\mathbf{p} \in D \mid p_h \leq u'\right] &= \mathbb{E}\left[\mathbb{P}[\mathbf{p} \in D \mid p_h] \mid p_h \leq u'\right] \\ &= \gamma\mathbb{E}[\mathbb{P}[\mathbf{p} \in D \mid p_h] \mid p_h \leq u] + (1-\gamma)\mathbb{E}\left[\mathbb{P}[\mathbf{p} \in D \mid p_h] \mid u < p_h \leq u'\right] \\ &\geq \mathbb{E}[\mathbb{P}[\mathbf{p} \in D \mid p_h] \mid p_h \leq u] = \mathbb{P}[\mathbf{p} \in D \mid p_h \leq u]\,.\end{aligned}$$

We can then apply Lemma 28 of Section 7 with $U = p_h$ and $V = |R|$. ■

6.2.1 PROOF OF THEOREM 22

By definition of a step-up procedure, the two-stage procedure R satisfies the assumption of Lemma 26 for $G(\mathbf{p}) = (1 - \frac{|R_0|}{m})^{-1}$, where R_0 is the first stage with FWER controlled at level α_0 . Furthermore, it is easy to check that $|R|$ is nonincreasing as a function of each p-value (since $|R_0|$ is). Then, we can apply Lemma 26, and from inequality (8) we deduce

$$\begin{aligned}FDR(R) &\leq \alpha_1 + \mathbb{E}\left[\frac{|R \cap \mathcal{H}_0|}{|R|}\mathbf{1}\left\{1 - \frac{|R_0|}{m} < \pi_0\right\}\right] \\ &\leq \alpha_1 + \mathbb{P}[R_0 \cap \mathcal{H}_0 \neq \emptyset] \\ &\leq \alpha_0 + \alpha_1\,.\end{aligned}$$

In the case where R_0 rejects all hypotheses, we assumed implicitly that the second stage also does. ■

6.2.2 PROOF OF THEOREM 23

Assume $\pi_0 > 0$ (otherwise the result is trivial). By definition of a step-up procedure, the two-stage procedure R satisfies the assumption of Lemma 26 for $G(\mathbf{p}) = F_\kappa(|R_0|/m)$, where R_0 is the first stage. Furthermore, it is easy to check that $|R|$ is nonincreasing as a function of each p-value (since $|R_0|$ is). Then, we can apply Lemma 26, and from inequality (8) we deduce

$$\begin{aligned}\mathrm{FDR}(R) &\leq \alpha_1 + \mathbb{E}\left[\frac{|R \cap \mathcal{H}_0|}{|R|}\mathbf{1}\left\{F_\kappa(|R_0|/m) > \pi_0^{-1}\right\}\right] \\ &\leq \alpha_1 + m_0\mathbb{E}\left[\frac{\mathbf{1}\left\{F_\kappa(|R_0|/m) > \pi_0^{-1}\right\}}{|R_0|}\right].\end{aligned}$$

For the second inequality, we have used the two following facts:

(i) $F_\kappa(|R_0|/m) > \pi_0^{-1}$ implies $|R_0| > 0$,

(ii) because of the assumption $\alpha_0 \leq \alpha_1$ and $F_\kappa \geq 1$, the output of the second step is necessarily a set containing at least the output of the first step. Hence $|R| \geq |R_0|$.

Let us now concentrate on further bounding this second term. For this, first consider the generalized inverse of F_κ, $F_\kappa^{-1}(t) = \inf\{x \mid F_\kappa(x) > t\}$. Since F_κ is a nondecreasing left-continuous function, we have $F_\kappa(x) > t \Leftrightarrow x > F_\kappa^{-1}(t)$. Furthermore, the expression of F_κ^{-1} is given by: $\forall t \in [1,+\infty), F_\kappa^{-1}(t) = \kappa^{-1}t^{-2} - t^{-1} + 1$ (providing in particular that $F_\kappa^{-1}(\pi_0^{-1}) > 1 - \pi_0$). Hence

$$m_0 \mathbb{E}\left[\frac{\mathbf{1}\left\{F_\kappa(|R_0|/m) > \pi_0^{-1}\right\}}{|R_0|}\right] \leq m_0 \mathbb{E}\left[\frac{\mathbf{1}\left\{|R_0|/m > F_\kappa^{-1}(\pi_0^{-1})\right\}}{|R_0|}\right] \leq \frac{\pi_0}{F_\kappa^{-1}(\pi_0^{-1})} \mathbb{P}\left[|R_0|/m \geq F_\kappa^{-1}(\pi_0^{-1})\right]. \tag{9}$$

Now, by assumption, the FDR of the first step R_0 is controlled at level $\pi_0\alpha_0$, so that

$$\begin{aligned}
\pi_0\alpha_0 &\geq \mathbb{E}\left[\frac{|R_0 \cap \mathcal{H}_0|}{|R_0|}\mathbf{1}\{|R_0| > 0\}\right] \\
&\geq \mathbb{E}\left[\frac{|R_0| + m_0 - m}{|R_0|}\mathbf{1}\{|R_0| > 0\}\right] \\
&= \mathbb{E}\left[\left(1 + (\pi_0 - 1)Z^{-1}\right)\mathbf{1}\{Z > 0\}\right],
\end{aligned}$$

where we denoted by Z the random variable $|R_0|/m$. Hence by Markov's inequality, for all $t > 1 - \pi_0$,

$$\mathbb{P}[Z \geq t] \leq \mathbb{P}\left[\left(1 + (\pi_0 - 1)Z^{-1}\right)\mathbf{1}\{Z > 0\} \geq 1 + (\pi_0 - 1)t^{-1}\right] \leq \frac{\pi_0\alpha_0}{1 + (\pi_0 - 1)t^{-1}};$$

choosing $t = F_\kappa^{-1}(\pi_0^{-1})$ and using this into (9), we obtain

$$m_0 \mathbb{E}\left[\frac{\mathbf{1}\left\{F_\kappa(|R_0|/m) > \pi_0^{-1}\right\}}{|R_0|}\right] \leq \alpha_0 \frac{\pi_0^2}{F_\kappa^{-1}(\pi_0^{-1}) - 1 + \pi_0}.$$

If we want this last quantity to be less than $\kappa\alpha_0$, this yields the condition $F_\kappa^{-1}(\pi_0^{-1}) \geq \kappa^{-1}\pi_0^2 - \pi_0 + 1$, and this is true from the expression of F_κ^{-1} (note that this is how the formula for F_κ was determined in the first place). ■

7. Probabilistic Lemmas

The three following lemmas have been established in a previous work (see Blanchard and Roquain, 2008, Lemma 3.2).

Lemma 27 *Let $g : [0,1] \to (0,\infty)$ be a nonincreasing function. Let U be a random variable which is stochastically lower bounded by a uniform variable on $[0,1]$, that is, $\forall u \in [0,1]$, $\mathbb{P}[U \leq u] \leq u$. Then, for any constant $c > 0$, we have*

$$\mathbb{E}\left[\frac{\mathbf{1}\{U \leq cg(U)\}}{g(U)}\right] \leq c.$$

Lemma 28 *Let U, V be two nonnegative real variables. Assume the following:*

1. *U is stochastically lower bounded by a uniform variable on $[0,1]$, that is, $\forall u \in [0,1]$, $\mathbb{P}[U \leq u] \leq u$.*
2. *The conditional distribution of V given $U \leq u$ is stochastically decreasing in u, that is,*

$$\forall v \geq 0, \qquad \forall 0 \leq u \leq u', \qquad \mathbb{P}[V < v \mid U \leq u] \leq \mathbb{P}\left[V < v \mid U \leq u'\right].$$

Then, for any constant $c > 0$, we have

$$\mathbb{E}\left[\frac{\mathbf{1}\{U \leq cV\}}{V}\right] \leq c.$$

Lemma 29 *Let U, V be two nonnegative real variables and β be a function of the form (3). Assume that U is stochastically lower bounded by a uniform variable on $[0,1]$, that is, $\forall u \in [0,1]$, $\mathbb{P}[U \leq u] \leq u$. Then, for any constant $c > 0$, we have*

$$\mathbb{E}\left[\frac{\mathbf{1}\{U \leq c\beta(V)\}}{V}\right] \leq c.$$

The following lemma was stated by Benjamini et al. (2006). It is a major point when we estimate π_0^{-1} in the independent case. The proof is left to the reader.

Lemma 30 *For any $k \geq 2$, $q \in (0,1]$, let Y be a binomial random variable with parameters $(k-1, q)$; then the following holds:*

$$\mathbb{E}[1/(1+Y)] \leq 1/kq.$$

Acknowledgments

The first author's work was carried out at the Fraunhofer Institute FIRST, Berlin. The second author's research was mostly carried out at the French institute INRA and at the VU University in Amsterdam, that are warmly acknowledged. Both authors would like to thank the Associated Editor and the two referees for their helpful comments and suggestions.

References

Y. Benjamini and R. Heller. False discovery rates for spatial signals. *Journal of the American Statistical Association*, 102(480):1272–1281, 2007.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Ser. B*, 57(1):289–300, 1995.

Y. Benjamini and Y. Hochberg. On the adaptive control of the false discovery rate in multiple testing with independent statistics. *Journal of Educational and Behavioral Statistics*, 25:60–83, 2000.

Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29(4):1165–1188, 2001.

Y. Benjamini, A. M. Krieger, and D. Yekutieli. Adaptive linear step-up procedures that control the false discovery rate. *Biometrika*, 93(3):491–507, 2006.

M. A. Black. A note on the adaptive control of false discovery rates. *Journal of the Royal Statistical Society, Series B*, 66(2):297–304, 2004.

G. Blanchard and F. Fleuret. Occam's hammer. In N.H. Bshouty and C. Gentile, editors, *Proceedings of the 20th. Conference on Learning Theory (COLT 2007)*, volume 4539 of *Springer Lecture Notes on Computer Science*, pages 112–126, 2007.

G. Blanchard and E. Roquain. Two simple sufficient conditions for FDR control. *Electronic Journal of Statistics*, 2:963–992, 2008.

S. Dudoit, J. Shaffer, and J. Boldrick. Multiple hypothesis testing in microarray experiments. *Statistical Science*, 18(1):71–103, 2003.

B. Efron, R. Tibshirani, J. Storey, and V. Tusher. Empirical Bayes analysis of a microarray experiment. *Journal of the American Statistical Association*, 96(456):1151–1160, 2001.

A. Farcomeni. Some results on the control of the false discovery rate under dependence. *Scandinavian Journal of Statistics*, 34(2):275–297, 2007.

H. Finner and M. Roters. On the false discovery rate and expected type I errors. *Biometrical Journal*, 43(8):985–1005, 2001.

H. Finner, T. Dickhaus, and M. Roters. Dependency and false discovery rate: Asymptotics. *Annals of Statistics*, 35(4):1432–1455, 2007.

H. Finner, T. Dickhaus, and M. Roters. On the false discovery rate and an asymptotically optimal rejection curve. *Annals of Statistics*, 37(2):596–618, 2009.

Y. Gavrilov, Y. Benjamini, and S. K. Sarkar. An adaptive step-down procedure with proven FDR control under independence. *Annals of Statistics*, 37(2):619–629, 2009.

C. Genovese and L. Wasserman. Operating characteristics and extensions of the false discovery rate procedure. *Journal of the Royal Statistical Society, Series B*, 64(3):499–517, 2002.

C. Genovese and L. Wasserman. A stochastic process approach to false discovery control. *Annals of Statistics*, 32(3):1035–1061, 2004.

C. Genovese, K. Roeder, and L. Wasserman. False discovery control with p-value weighting. *Biometrika*, 93(3):509–524, 2006.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

J. Jin. Proportion of non-zero normal means: universal oracle equivalences and uniformly consistent estimators. *Journal of the Royal Statistical Society, Series B*, 70(3):461–493, 2008.

J. Jin and T. Cai. Estimating the null and the proportion of nonnull effects in large-scale multiple comparisons. *Journal of the American Statistical Association*, 102(478):495–506, 2007.

E. L. Lehmann. Some concepts of dependence. *Annals of Mathematical Statistics*, 37:1137–1153, 1966.

N. Meinshausen and J. Rice. Estimating the proportion of false null hypotheses among a large number of independently tested hypotheses. *Annals of Statistics*, 34(1):373–393, 2006.

P. Neuvial. Asymptotic properties of false discovery rate controlling procedures under independence. *Electronic Journal of Statistics*, 2:1065–1110, 2008.

E. Roquain. *Exceptional Motifs in Heterogeneous Sequences. Contributions to Theory and Methodology of Multiple Testing*. PhD thesis, Université Paris XI, 2007.

E. Roquain and M. van de Wiel. Optimal weighting for false discovery rate control. *Electronic Journal of Statistics*, 3:678–711, 2009.

S. K. Sarkar. Two-stage stepup procedures controlling FDR. *Journal of Statistical Planning and Inference*, 138(4):1072–1084, 2008a.

S. K. Sarkar. On methods controlling the false discovery rate. *Sankhya, Series A*, 70:135–168, 2008b.

C. Scott and G. Blanchard. Novelty detection: unlabaled data definitely help. In D. van Dyk and M. Welling, editors, *Proceedings of the 12th. International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, volume 5 of *JMLR Workshop and Conference Proceedings*, pages 464–471, 2009.

E. Spjøtvoll. On the optimality of some multiple comparison procedures. *Annals of Mathematical Statistics*, 43(2):398–411, 1972.

J. Storey. A direct approach to false discovery rates. *Journal of the Royal Statistical Society, Series B*, 64(3):479–498, 2002.

J. Storey. The optimal discovery procedure: a new approach to simultaneous significance testing. *Journal of the Royal Statistical Society, Series B*, 69(3):347–368, 2007.

J. Storey and R. Tibshirani. SAM thresholding and false discovery rates for detecting differential gene expression in DNA microarrays. In *The Analysis of Gene Expression Data*, Statistics for Biology and Health, pages 272–290. Springer, New York, 2003.

J. Storey, J. Taylor, and D. Siegmund. Strong control, conservative point estimation and simultaneous conservative consistency of false discovery rates: a unified approach. *Journal of the Royal Statistical Society, Series B*, 66(1):187–205, 2004.

W. Sun and T. Cai. Oracle and adaptive compound decision rules for false discovery rate control. *Journal of the American Statistical Association*, 102(479):901–912, 2007.

L. Wasserman and K. Roeder. Weighted hypothesis testing. Technical report 830, Dept. of statistics, Carnegie Mellon University, 2006. ArXiv preprint math/0604172v1. URL: `http://arxiv.org/abs/math.ST/0604172`.

Journal of Machine Learning Research 10 (2009) 2873-2898 Submitted 1/08; Revised 7/09; Published 12/09

Online Learning with Samples Drawn from Non-identical Distributions

Ting Hu TING_HU2005@163.COM
Ding-Xuan Zhou MAZHOU@CITYU.EDU.HK
Department of Mathematics
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong, China

Editor: Peter Bartlett

Abstract

Learning algorithms are based on samples which are often drawn independently from an identical distribution (i.i.d.). In this paper we consider a different setting with samples drawn according to a non-identical sequence of probability distributions. Each time a sample is drawn from a different distribution. In this setting we investigate a fully online learning algorithm associated with a general convex loss function and a reproducing kernel Hilbert space (RKHS). Error analysis is conducted under the assumption that the sequence of marginal distributions converges polynomially in the dual of a Hölder space. For regression with least square or insensitive loss, learning rates are given in both the RKHS norm and the L^2 norm. For classification with hinge loss and support vector machine q-norm loss, rates are explicitly stated with respect to the excess misclassification error.

Keywords: sampling with non-identical distributions, online learning, classification with a general convex loss, regression with insensitive loss and least square loss, reproducing kernel Hilbert space

1. Introduction

In the literature of learning theory, samples for algorithms are often assumed to be drawn independently from an identical distribution. Here we consider a setting with samples drawn from non-identical distributions. Such a framework was introduced in Smale and Zhou (2009) and Steinwart et al. (2008) where online learning for least square regression and off-line support vector machines are investigated. We shall follow this framework and study a kernel based online learning algorithm associated with a general convex loss function. Our analysis can be applied for various purposes including regression and classification.

1.1 Sampling with Non-identical Distributions

Let (X,d) be a metric space called an input space for the learning problem. Let Y be a compact subset of $\mathbb{R}$ (output space) and $Z = X \times Y$.

In our online learning setting, at each step $t = 1,2,\ldots$, a pair $z_t = (x_t, y_t)$ is drawn from a probability distribution $\rho^{(t)}$ on Z. The sampling sequence of probability distributions $\{\rho^{(t)}\}_{t=1,2,\cdots}$ is not identical. For convergence analysis, we shall assume that the sequence $\{\rho_X^{(t)}\}_{t=1,2,\cdots}$ of marginal distributions on X converges polynomially in the dual of the Hölder space $C^s(X)$ for some $0 < s \leq 1$.

Here the Hölder space $C^s(X)$ is defined to be the space of all continuous functions on X with the norm $\|f\|_{C^s(X)} = \|f\|_{C(X)} + |f|_{C^s(X)}$ finite, where $|f|_{C^s(X)} := \sup_{x\neq y} \frac{|f(x)-f(y)|}{(d(x,y))^s}$.

Definition 1 *We say that the sequence $\{\rho_X^{(t)}\}_{t=1,2,\cdots}$ converges polynomially to a probability distribution ρ_X in $(C^s(X))^*$ ($0 \le s \le 1$) if there exist $C > 0$ and $b > 0$ such that*

$$\|\rho_X^{(t)} - \rho_X\|_{(C^s(X))^*} \le Ct^{-b}, \qquad t \in \mathbb{N}. \tag{1}$$

By the definition of the dual space $(C^s(X))^*$, decay condition (1) can be expressed as

$$\left| \int_X f(x) d\rho_X^{(t)} - \int_X f(x) d\rho_X \right| \le Ct^{-b} \|f\|_{C^s(X)}, \qquad \forall f \in C^s(X), t \in \mathbb{N}. \tag{2}$$

What measures quantitatively differences between our non-identical setting and the i.i.d. case is the power index b. Its impact on performance of online learning algorithms will be studied in this paper. The i.i.d. case corresponds to $b = \infty$.

We describe three situations in which decay condition (1) is satisfied. The first is when a distribution ρ_X is perturbed by some noise and the noise level decreases as t increases.

Example 1 *Let $\{h^{(t)}\}$ be a sequence of bounded functions on X such that $\sup_{x\in X} |h^{(t)}(x)| \le Ct^{-b}$. Then the sequence $\{\rho_X^{(t)}\}_{t=1,2,\cdots}$ defined by $d\rho_X^{(t)} = d\rho_X + h^{(t)}(x)d\rho_X$ satisfies (1) for any $0 \le s \le 1$.*

The proof follows from $\left|\int_X f(x)h^{(t)}(x)d\rho_X\right| \le \sup_{x\in X} |h^{(t)}(x)| \|f\|_{C(X)} \le Ct^{-b}\|f\|_{C^s(X)}$. In this example, $h^{(t)}$ is the density function of the noise distribution and we assume its bound (noise level) to decay polynomially as t increases.

The second situation when decay condition (1) is satisfied is generated by iterative actions of an integral operator associated with a stochastic density kernel. We demonstrate this situation by an example on $X = S^{n-1}$, the unit sphere of $\mathbb{R}^n$ with $n \ge 2$. Let dS be the normalized surface element of S^{n-1}. The corresponding space $L^2(S^{n-1})$ has an orthonormal basis $\{Y_{\ell,k} : \ell \in \mathbb{Z}_+, k = 1, \ldots, N(n,\ell)\}$ with $N(n,0) = 1$ and $N(n,\ell) = \frac{2\ell+n-2}{\ell} \frac{(\ell+n-3)!(n-2)!}{(\ell-1)!}$. Here $Y_{\ell,k}$ is a spherical harmonic of order ℓ which is the restriction onto S^{n-1} of a homogeneous polynomial in $\mathbb{R}^n$ of degree ℓ satisfying the Laplace equation $\Delta f = 0$. In particular, $Y_{0,0} \equiv 1$.

Example 2 *Let $X = S^{n-1}$, $0 < \alpha < 1$, and $\psi \in C(X \times X)$ be given by*

$$\psi(x,u) = 1 + \sum_{\ell=1}^{\infty} \sum_{k=1}^{N(n,\ell)} a_{\ell,k} Y_{\ell,k}(x) Y_{\ell,k}(u) \quad \textit{where} \quad 0 \le a_{\ell,k} \le \alpha, \ \sum_{\ell=1}^{\infty} \sum_{k=1}^{N(n,\ell)} a_{\ell,k} \|Y_{\ell,k}\|_{C(X)}^2 < 1.$$

If $h^{(1)}$ is a square integrable density function on X and a sequence of density functions $\{h^{(t)}\}$ is defined by

$$h^{(t+1)}(x) = \int_X \psi(x,u) h^{(t)}(u) dS(u), \qquad x \in X, \ t \in \mathbb{N},$$

then we know $h^{(t)} = Y_{0,0} + \sum_{\ell=1}^{\infty} \sum_{k=1}^{N(n,\ell)} a_{\ell,k}^{t-1} \langle h^{(1)}, Y_{\ell,k} \rangle_{L^2(S^{n-1})} Y_{\ell,k}$ and $\|h^{(t)} - Y_{0,0}\|_{L^2(S^{n-1})} \le \alpha^{t-1} \|h^{(1)}\|_{L^2(S^{n-1})}$. It follows that the sequence $\{\rho_X^{(t)} = h^{(t)}(x)dS\}_{t=1,2,\cdots}$ of probability distributions on X converges polynomially to the uniform distribution $d\rho_X = dS$ on X and satisfies (1) for any $0 \le s \le 1$.

In general, if ν is a strictly positive probability distribution on X, and if $\psi \in C(X \times X)$ is strictly positive satisfying $\int_X \psi(x,u)d\nu(u) = 1$ for each $x \in X$, then the sequence $\{\rho_X^{(t)}\}$ defined by

$$\rho_X^{(t)}(\Gamma) = \int_\Gamma \left\{ \int_X \psi(x,u) d\rho_X^{(t-1)}(x) \right\} d\nu(u) \qquad \text{on Borel sets} \quad \Gamma \subseteq X$$

satisfies $\|\rho_X^{(t)} - \rho_X\|_{(C(X))^*} \leq C\alpha^t$ for some (strictly positive) probability distribution ρ_X on X and constants $C > 0, 0 < \alpha < 1$. Hence decay condition (1) is valid for any $0 \leq s \leq 1$. For details, see Smale and Zhou (2009).

The third situation to realize decay condition (1) is to induce distributions by dynamical systems. Here we present a simple example.

Example 3 *Let $X = [-1/2, 1/2]$ and for each $t \in \mathbb{N}$, the probability distribution $\rho_X^{(t)}$ on X has support $[-2^{-t}, 2^{-t}]$ and uniform density 2^{t-1} on its support. Then with δ_0 being the delta distribution at the origin, for each $0 < s \leq 1$ we have*

$$\left| \int_X f(x) d\rho_X^{(t)} - \int_X f(x) d\delta_0 \right| \leq 2^{t-1} \int_{-2^{-t}}^{2^{-t}} |f(x) - f(0)| dx \leq \left(2^{-s}\right)^t \|f\|_{C^s(X)}.$$

Remark 2 *Since $\|f\|_{C(X)} \leq \|f\|_{C^s(X)}$, we see from (2) that decay condition (1) with any $0 < s \leq 1$ is satisfied when this polynomial convergence requirement is valid in the case $s = 0$. This happens in Examples 1 and 2. Note that when $s = 0$, the dual space $(C(X))^*$ is exactly the space of signed finite measures on X. Each signed finite measure μ on X lies in $(C(X))^* \subset (C^s(X))^*$ and satisfies $\|\mu\|_{(C^s(X))^*} \leq \|\mu\|_{(C(X))^*} \leq \int_X d|\mu|$.*

1.2 Fully Online Learning Algorithm

In this paper we study a family of online learning algorithms associated with reproducing kernel Hilbert spaces and a general convex loss function.

A *reproducing kernel Hilbert space* (RKHS) is induced by a *Mercer kernel* $K : X \times X \to \mathbb{R}$ which is a continuous and symmetric function such that the matrix $(K(x_i, x_j))_{i,j=1}^\ell$ is positive semidefinite for any finite set of points $\{x_1, \cdots, x_\ell\} \subset X$. The RKHS $\mathcal{H}_K$ is the completion (Aronszajn, 1950) of the span of the set of functions $\{K_x = K(x, \cdot) : x \in X\}$ with the inner product given by $\langle K_x, K_y \rangle_K = K(x,y)$.

Definition 3 *We say that $V : Y \times \mathbb{R} \to \mathbb{R}_+$ is a convex loss function if for each $y \in Y$, the univariate function $V(y, \cdot) : \mathbb{R} \to \mathbb{R}_+$ is convex.*

The convexity tells us (Rockafellar, 1970) that for each $f \in \mathbb{R}$ and $y \in Y$, the left derivative $\lim_{\delta \to 0-}(V(y, f+\delta) - V(y,f))/\delta$ exists and is no more than the right derivative $\lim_{\delta \to 0+}(V(y, f+\delta) - V(y,f))/\delta$. An arbitrary number between them (which is a gradient) will be taken and denoted as $\partial V(y,f)$ in our algorithm.

For the least square regression problem, we can take $V(y,f) = (y-f)^2$. For the binary classification problem, we can take $V(y,f) = \phi(yf)$ with $\phi : \mathbb{R} \to \mathbb{R}_+$ a convex function.

The online algorithm associated with the RKHS $\mathcal{H}_K$ and the convex loss V is a stochastic gradient descent method (Cesa-Bianchi et al., 1996; Kivinen et al., 2004; Smale and Yao, 2006; Ying and Zhou, 2006; Ying, 2007).

Definition 4 *The fully online learning algorithm is defined by* $f_1 = 0$ *and*

$$f_{t+1} = f_t - \eta_t \{\partial V(y_t, f_t(x_t))K_{x_t} + \lambda_t f_t\}, \text{ for } t = 1, 2, \ldots, \tag{3}$$

where $\lambda_t > 0$ *is called the regularization parameter and* $\eta_t > 0$ *the step size.*

In this fully online algorithm, the regularization parameter λ_t changes with the learning step t. Throughout the paper we assume that $\lambda_{t+1} \leq \lambda_t$ for each $t \in \mathbb{N}$. When the regularization parameter $\lambda_t \equiv \lambda_1$ does not change as the step t develops, we call scheme (3) *partially online*.

The goal of this paper is to investigate the fully online learning algorithm (3) when the sampling sequence is not identical. We will show that learning rates in the non-identical setting can be the same as those in the i.i.d. case when the power index b in polynomial decay condition (1) is large enough, that is, $\{\rho_X^{(t)}\}$ converges fast to ρ_X. When b is small, the non-identical effect becomes crucial and the learning rates will depend essentially on b.

2. Error Bounds for Regression and Classification

As in the work on least square regression (Smale and Zhou, 2009), we assume for the sampling sequence $\{\rho^{(t)}\}_{t=1,2,\cdots}$ that the conditional distribution $\rho^{(t)}(y|x)$ of each $\rho^{(t)}$ at $x \in X$ is independent of t, denoted as ρ_x.

Throughout the paper we assume independence of the sampling, that is, $\{z_t = (x_t, y_t)\}_t$ is a sequence of samples drawn from the product probability space $\Pi_{t=1,2,\cdots}(Z, \rho^{(t)})$.

Error analysis will be conducted for fully online learning algorithm (3) under polynomial decay condition (1) for the sequence of marginal distributions $\{\rho_X^{(t)}\}$. Let ρ be the probability distribution on Z given by the marginal distribution ρ_X and the conditional distributions ρ_x. Essential difficulty in our non-identical setting is caused by the deviation of $\{\rho^{(t)}\}$ from ρ.

The first novelty of our analysis is to deal with an error quantity Δ_t involving $\rho^{(t)} - \rho$ (defined by (15) below) which occurs only in the non-identical setting. This is handled for a general loss function V and output space Y by Lemma 18 in Section 3 under decay condition (1) for marginal distributions $\{\rho_X^{(t)}\}$ and Lipschitz s continuity of conditional distributions $\{\rho_x : x \in X\}$.

Definition 5 *We say that the set of distributions* $\{\rho_x : x \in X\}$ *is Lipschitz* s *in* $(C^s(Y))^*$ *if there exists a constant* $C_\rho \geq 0$ *such that*

$$\|\rho_x - \rho_u\|_{(C^s(Y))^*} \leq C_\rho (d(x,u))^s, \qquad \forall x, u \in X. \tag{4}$$

Notice that on the compact subset Y of $\mathbb{R}$, the Hölder space $C^s(Y)$ and its dual $(C^s(Y))^*$ are well defined. Each ρ_x belongs to $(C^s(Y))^*$.

The second novelty of our analysis is to show for the least square loss (described in Section 3) and binary classification that Lipschitz s continuity (4) of $\{\rho_x : x \in X\}$ is the same as requiring $f_\rho \in C^s(X)$ where f_ρ is the *regression function* defined by

$$f_\rho(x) = \int_Y y d\rho_x(y), \qquad x \in X. \tag{5}$$

Proposition 6 *Let* $0 < s \leq 1$. *Condition (4) implies* $f_\rho \in C^s(X)$ *with* $|f_\rho|_{C^s(X)} \leq C_\rho(1 + 2^{1-s}) \sup_{y \in Y} |y|$. *When* $Y = \{1, -1\}$, $f_\rho \in C^s(X)$ *also implies (4) and* $C_\rho \leq |f_\rho|_{C^s(X)}$.

The two-point nature of the output space Y for binary classification plays a crucial role in our observation. The second statement of Proposition 6 is not true for general output space Y. Here is one example.

Example 4 *Let $0 < s \leq 1$ and $Y = \{1, -1, 0\}$. Then condition (4) holds if and only if $f_\rho \in C^s(X)$ and $f_{\rho,-1} \in C^s(X)$ where $f_{\rho,-1}$ is the function on X given by $f_{\rho,-1}(x) = \rho_x(\{-1\})$.*

Proofs of Proposition 6 and Example 4 will be given in the appendix.

Our third novelty is to understand some essential differences between our non-identical setting and the classical i.i.d. setting by pointing out the key role played by the power index b of polynomial decay condition $\|\rho_X^{(t)} - \rho_X\|_{(C^s(X))^*} \leq Ct^{-b}$ in derived convergence rates in Theorems 7 and 10 for regression and Theorem 11 for classification. Even for least square regression our result improves the error analysis in Smale and Zhou (2009) where a stronger exponential decay condition $\|\rho_X^{(t)} - \rho_X\|_{(C^s(X))^*} \leq C\alpha^t$ is assumed.

Our error bounds for fully online algorithm (3) are comparable with those for a batch learning algorithm generated by the off-line regularization scheme in $\mathcal{H}_K$ defined with a sample $\mathbf{z} := \{z_t = (x_t, y_t)\}_{t=1}^T$ and a regularization parameter $\lambda > 0$ as

$$f_{\mathbf{z},\lambda} = \arg\min_{f \in \mathcal{H}_K} \left\{ \frac{1}{T} \sum_{t=1}^{T} V(y_t, f(x_t)) + \frac{\lambda}{2} \|f\|_K^2 \right\}. \tag{6}$$

Let us demonstrate our error analysis by learning rates for regression with least square loss and insensitive loss and for binary classification with hinge loss.

2.1 Learning Rates for Least Square Regression

Here we take $Y = [-M, M]$ for some $M > 0$ and the least square loss $V = V_{ls}$ as $V_{ls}(y, f) = (y - f)^2$. Then the algorithm takes the form

$$f_{t+1} = f_t - 2\eta_t \left\{ (f_t(x_t) - y_t) K_{x_t} + \frac{\lambda_t}{2} f_t \right\}, \text{ for } t = 1, 2, \ldots$$

The following learning rates are derived by the procedure in Smale and Zhou (2009) where an exponential decay condition is assumed. Here we only impose a much weaker polynomial decay condition (1). We also assume the regularity condition (of order $r > 0$)

$$f_\rho = L_K^r(g_\rho) \text{ for some } g_\rho \in L^2_{\rho_X}(X), \tag{7}$$

where L_K is the integral operator $L^2_{\rho_X}$ defined by

$$L_K f(x) = \int_X K(x, v) f(v) d\rho_X(v), \qquad x \in X$$

with L_K^r well-defined as a compact operator.

Theorem 7 *Let $0 < s \leq \frac{1}{2}$ and $\frac{1}{2} < r \leq \frac{3}{2}$. Assume $K \in C^{2s}(X \times X)$, regularity condition (7) for f_ρ and (1) with $b > \frac{2r+2}{2r+1}$ for $\{\rho_X^{(t)}\}$. Take*

$$\lambda_t = \lambda_1 t^{-\frac{1}{2r+1}}, \quad \eta_t = \eta_1 t^{-\frac{2r}{2r+1}}$$

with $\lambda_1\eta_1 > \frac{2r-1}{4r+2}$, *then*

$$\mathbb{E}_{z_1,\ldots,z_T}\left(\|f_{T+1}-f_\rho\|_K\right) \le \widetilde{C}T^{-\frac{2r-1}{4r+2}},$$

where $\widetilde{C}$ *is a constant independent of* T.

Denote the constant $\kappa = \max_{x\in X}\sqrt{K(x,x)}$. From the reproducing property

$$\langle K_x, f\rangle_K = f(x), \qquad x\in X,\ f\in\mathcal{H}_K \tag{8}$$

of the RKHS $\mathcal{H}_K$, we see that

$$\|f\|_{C(X)} \le \kappa\|f\|_K, \qquad \forall x\in X, f\in\mathcal{H}_K.$$

Most error analysis in the literature of least square regression (Zhang, 2004; De Vito et al., 2005; Smale and Yao, 2006; Wu et al., 2007) is about the L^2-norm $\|f_{T+1}-f_\rho\|_{L^2_{\rho_X}}$ or risk in the i.i.d. case. From a predictive viewpoint, in the non-identical setting, the error $f_{T+1}-f_\rho$ should be measured with respect to the distribution $\rho_X^{(T)}$, not the limit ρ_X. This can be done by bounding $\|f_{T+1}-f_\rho\|_{C(X)}$ (since $\rho_X^{(T)}$ changes with T), which follows from estimates for $\|f_{T+1}-f_\rho\|_K$. So our bounds for the error in the $\mathcal{H}_K$-norm provides useful predictive information about learning ability of fully online algorithm (3) in the non-identical setting.

Remark 8 *When* $X\subset\mathbb{R}^n$ *and* $K\in C^{2m}(X\times X)$ *for some* $m\in\mathbb{N}$, *we know from Zhou (2003, 2008) and Theorem 7 that* $\mathbb{E}_{z_1,\ldots,z_T}\left(\|f_{T+1}-f_\rho\|_{C^m(X)}\right) = O(T^{-\frac{2r-1}{4r+2}})$. *So the regression function is learned efficiently by the online algorithm not only in the usual* $L^2_{\rho_X}$ *space, but also strongly in the space* $C^m(X)$ *implying the learning of gradients (Mukherjee and Wu, 2006).*

In the special case of $r=\frac{3}{2}$, the learning rate in Theorem 7 is $\mathbb{E}_{z_1,\ldots,z_T}\left(\|f_{T+1}-f_\rho\|_K\right) = O(T^{-\frac{1}{4}})$, the same as those in the literature (Smale and Zhou, 2009; Tarrès and Yao, 2005; Smale and Zhou, 2007). Here we assume polynomial convergence condition (1) with a large index $b > \frac{2r+2}{2r+1}$. So the influence of the non-identical distributions $\{\rho_X^{(t)}\}$ does not appear in the learning rates (it is involved in the constant $\widetilde{C}$). Instead of refining the analysis for smaller b in Theorem 7, we shall show the influence of the index b on learning rates by the settings of regression with insensitive loss and binary classification.

2.2 Learning Rates for Regression with Insensitive Loss

A large family of loss functions for regression take the form $V(y,f)=\psi(y-f)$ where $\psi:\mathbb{R}\to\mathbb{R}_+$ is an even, convex and continuous function satisfying $\psi(0)=0$. One example is the ε-insensitive loss (Vapnik, 1998) with $\varepsilon\ge 0$ where $\psi(u)=\max\{|u|-\varepsilon,0\}$. We consider the case when $\varepsilon=0$. In this case the loss is called least absolute deviation or least absolute error in the literature of statistics and finds applications in some important problems because of robustness.

Definition 9 *The insensitive loss* $V=V_{in}$ *is given by* $V_{in}(y,f)=|y-f|$.

Algorithm (3) now takes the form

$$f_{t+1} = \begin{cases} (1-\eta_t\lambda_t)f_t - \eta_t K_{x_t}, & \text{if } f_t(x_t)\ge y_t,\\ (1-\eta_t\lambda_t)f_t + \eta_t K_{x_t}, & \text{if } f_t(x_t)< y_t. \end{cases}$$

The following learning rates are new and will be proved in Section 5.

Theorem 10 *Let $0 < s \le \frac{1}{2}$ and $K \in C^{2s}(X \times X)$. Assume regularity condition (7) for f_ρ with $r > \frac{1}{2}$, and polynomial convergence condition (1) for $\{\rho_X^{(t)}\}$. Suppose that for each $x \in X$, ρ_x is the uniform distribution on the interval $[f_\rho(x) - 1, f_\rho(x) + 1]$. If $\lambda_1 \le (\kappa \|g_\rho\|_{L^2_{\rho_X}})^{2/(1-2r)}/2$ and $\eta_1 > 0$, then with a constant $\widetilde{C}$ independent of T, when $1 < r \le \frac{3}{2}$ we have*

$$\mathbb{E}_{z_1,\dots,z_T}\left(\|f_{T+1} - f_\rho\|_K\right) \le \widetilde{C} T^{-\min\{\frac{2r-1}{6r+1}, \frac{b}{2} - \frac{2}{6r+1}\}} \quad \text{by taking } \lambda_t = \lambda_1 t^{-\frac{2}{6r+1}},\ \eta_t = \eta_1 t^{-\frac{4r}{6r+1}}$$

and when $\frac{1}{2} < r \le 1$, we have

$$\mathbb{E}_{z_1,\dots,z_T}\left(\|f_{T+1} - f_\rho\|_{L^2_{\rho_X}}\right) \le \widetilde{C} T^{-\min\{\frac{r}{3r+2}, \frac{b}{2} - \frac{1}{3r+2}\}} \quad \text{with } \lambda_t = \lambda_1 t^{-\frac{1}{3r+2}},\ \eta_t = \eta_1 t^{-\frac{2r+1}{3r+2}}.$$

Again, when $b > \frac{4r+2}{6r+1}$, $X \subset \mathbb{R}^n$ and $K \in C^{2m}(X \times X)$ for some $m \in \mathbb{N}$, Theorem 10 tells us that $\mathbb{E}_{z_1,\dots,z_T}\left(\|f_{T+1} - f_\rho\|_{C^m(X)}\right) = O(T^{-\frac{2r-1}{6r+1}})$. So the regression function is learned efficiently by the online algorithm not only with respect to the risk, but also strongly in the space $C^m(X)$ implying the learning of gradients.

Consider the case $r = \frac{3}{2}$. When $\{\rho_X^{(i)}\}$ converges slowly with $b < \frac{4}{5}$, we see that $\mathbb{E}_{z_1,\dots,z_T}\left(\|f_{T+1} - f_\rho\|_K\right) = O(T^{-(\frac{b}{2} - \frac{1}{5})})$ which heavily depends on the index b representing quantitatively the deviation of $\{\rho_X^{(t)}\}$ from ρ_X. When b is large enough with $b \ge \frac{4}{5}$, the learning rate $\mathbb{E}_{z_1,\dots,z_T}\left(\|f_{T+1} - f_\rho\|_K\right)$ is of order $T^{-\frac{1}{5}}$ which is independent of b.

It would be interesting to extend Theorem 10 to situations when $\{\rho_x : x \in X\}$ are more general bounded symmetric distributions.

2.3 Learning Rates for Binary Classification

The output space Y for the binary classification problem is $Y = \{1, -1\}$ representing the set of two classes. A binary *classifier* $C : X \to Y$ makes a prediction $y = C(x) \in Y$ for each point $x \in X$.

A real valued function $f : X \to \mathbb{R}$ can be used to generate a classifier $C(x) = \operatorname{sgn}(f(x))$ where $\operatorname{sgn}(f(x)) = 1$ if $f(x) \ge 0$ and $\operatorname{sgn}(f(x)) = -1$ if $f(x) < 0$. A classifying convex loss $\phi : \mathbb{R} \to \mathbb{R}_+$ is often used for the real valued function f, to measure the local error $\phi(yf(x))$ suffered from the use of $\operatorname{sgn}(f)$ as a model for the process producing y at $x \in X$. Take $V(y, f) = \phi(yf)$ in out setting. Off-line classification algorithm (6) has been extensively studied in the literature. In particular, the error analysis is well done when the sample $\mathbf{z}$ is assumed to be an identical and independent drawer from a probability measure ρ on Z. See, for example, Evgeniou et al. (2000), Steinwart (2002), Zhang (2004) and Wu et al. (2007). Some analysis for off-line support vector machines with dependent samples can be found in Steinwart et al. (2008).

When the sample size T is very large, algorithm (6) might be practically challenging. Then online learning algorithms can be applied, which provide more efficient methods for classification. These algorithms are generalizations of the perceptron which has a long history. The error analysis for online algorithm (3) has also been conducted for classification in the i.i.d. setting, see, for example, Cesa-Bianchi et al. (2004), Ying and Zhou (2006), Ying (2007) and Ye and Zhou (2007).

Here we are interested in the error analysis for fully online algorithm (3) in the non-identical setting. The error is measured by the *misclassification error* $\mathcal{R}(C)$ defined to be the probability of the event $\{C(x) \ne y\}$ for a classifier C

$$\mathcal{R}(C) = \int_X \rho_x(y \ne C(x)) d\rho_X(x) = \int_X \rho_x(\{-C(x)\}) d\rho_X(x).$$

The best classifier minimizing the misclassification error is called the *Bayes rule* (e.g., Devroye et al. 1997) and can be expressed as $f_c = \mathrm{sgn}(f_\rho)$.

We are interested in the classifier $\mathrm{sgn}(f_{T+1})$ produced by the real valued function f_{T+1} using fully online learning algorithm (3). So our error analysis aims at the *excess misclassification error* $\mathcal{R}(\mathrm{sgn}(f_{T+1})) - \mathcal{R}(f_c)$.

We demonstrate our error analysis by a result, proved in Section 5, for the *hinge loss* $\phi(x) = (1-x)_+ = \max\{1-x, 0\}$. For this loss, the online algorithm (3) can be expressed as $f_1 = 0$ and

$$f_{t+1} = \begin{cases} (1-\eta_t\lambda_t)f_t, & \text{if } y_t f_t(x_t) > 1, \\ (1-\eta_t\lambda_t)f_t + \eta_t y_t K_{x_t}, & \text{if } y_t f_t(x_t) \le 1. \end{cases}$$

Theorem 11 *Let $V(y,f) = (1-yf)_+$ and $K \in C^{2s}(X \times X)$ for some $0 < s \le \frac{1}{2}$. Assume $f_\rho \in C^s(X)$ and the triple (ρ_X, f_c, K) satisfies*

$$\inf_{f \in \mathcal{H}_K} \{\|f - f_c\|_{L^1_{\rho_X}} + \frac{\lambda}{2}\|f\|_K^2\} \le \mathcal{D}_0 \lambda^\beta \qquad \forall \lambda > 0 \tag{9}$$

for some $0 < \beta \le 1$ and $\mathcal{D}_0 > 0$. Take

$$\lambda_t = \lambda_1 t^{-\frac{1}{4}}, \eta_t = \eta_1 t^{-(\frac{1}{2}+\frac{\beta}{12})}$$

where $\lambda_1 > 0$ and $0 < \eta_1 \le \frac{1}{2\kappa^2+\lambda_1}$. If $\{\rho_X^{(t)}\}_{t=1,2,\cdots}$ satisfies (1) with $b > \frac{1}{2}$, then

$$\mathbb{E}_{z_1,\ldots,z_T}\left(\mathcal{R}(sgn(f_{T+1})) - \mathcal{R}(f_c)\right) \le C_{\beta,s,b} T^{-\min\{\frac{\beta}{4}, \frac{1}{8}+\frac{\beta}{24}, \frac{b}{2}-\frac{1}{4}\}}, \tag{10}$$

where $C_{\beta,s,b} = C_{\eta_1,\lambda_1,\kappa,\mathcal{D}_0,\beta,s}$ is a constant depending on $\eta_1, \lambda_1, \kappa, \mathcal{D}_0, \beta, s$ and b.

In the i.i.d. case with $b = \infty$, the learning rate for fully online algorithm (3) we state in Theorem 11 is of form $O(T^{-\min\{\frac{\beta}{4}, \frac{1}{8}+\frac{\beta}{24}\}})$ which is better than that in Ye and Zhou (2007) of order $O(T^{-\min\{\frac{\beta}{4}, \frac{1}{8}-\frac{\varepsilon}{2}\}})$ with an arbitrarily small $\varepsilon > 0$. This improvement is realized by technical novelty in our error analysis, as pointed out in Remark 27. So even in the i.i.d. case, our learning rate for fully online classification with the hinge loss under approximation error assumption (9) is the best in the literature.

Let us discuss the role of the power index b for the convergence of $\{\rho_X^{(t)}\}$ to ρ_X played in the learning rate in Theorem 11. Consider the case $\beta \le \frac{3}{5}$. When $b \ge \frac{1+\beta}{2}$ meaning fast convergence of $\{\rho_X^{(t)}\}$ to ρ_X, learning rate (10) takes the form $O(T^{-\frac{\beta}{4}})$ which depends only on the approximation ability of $\mathcal{H}_K$ with respect to the function f_c and ρ_X. When $\frac{1}{2} < b < \frac{1+\beta}{2}$ representing some slow convergence of $\{\rho_X^{(t)}\}$ to ρ_X, the learning rate takes the form $O(T^{-\frac{2b-1}{4}})$ which depends only on b.

When $\beta > \frac{3}{5}$, learning rate (10) is of order $T^{-(\frac{1}{8}+\frac{\beta}{24})}$ for $b > \frac{3}{4} + \frac{\beta}{12}$ (fast convergence of $\{\rho_X^{(t)}\}$) and of order $T^{-(\frac{b}{2}-\frac{1}{4})}$ for $\frac{1}{2} < b \le \frac{3}{4} + \frac{\beta}{12}$ (slow convergence of $\{\rho_X^{(t)}\}$). It would be interesting to know how online learning algorithms adapt when the time dependent distribution drifts sufficiently slowly (corresponding to very small b).

2.4 Approximation Error Involving a General Loss

Condition (9) concerns the approximation of the function f_c in the space $L^1_{\rho_X}$ by functions from the RKHS $\mathcal{H}_K$. In particular, when $\mathcal{H}_K$ is a dense subset of $C(X)$ (i.e., K is a universal kernel), the quantity on the left-hand side of (9) tends to 0 as $\lambda \to 0$. So (9) is a reasonable assumption, which can be characterized by an interpolation space condition (Smale and Zhou, 2003; Chen et al., 2004).

Assumptions like (9) are necessary to determine the regularization parameter for achieving learning rate (10). This can be seen from the literature (Wu et al., 2007; Zhang, 2004; Caponnetto et al., 2007) of off-line algorithm (6): learning rates are obtained by suitable choices of the regularization parameter $\lambda \equiv \lambda_1 = \lambda_1(T)$, according to the behavior of the approximation error estimated from a priori conditions on the distribution ρ and the space $\mathcal{H}_K$.

For a general loss function V, conditions like (9) can be stated as the approximation error or regularization error.

Definition 12 *The approximation error $\mathcal{D}(\lambda)$ associated with the triple (K,V,ρ) is*

$$\mathcal{D}(\lambda) = \inf_{f\in\mathcal{H}_K} \left\{ \mathcal{E}(f) - \mathcal{E}(f_\rho^V) + \frac{\lambda}{2}\|f\|_K^2 \right\}, \tag{11}$$

where we define the generalization error for $f : X \to \mathbb{R}$ as

$$\mathcal{E}(f) = \int_Z V(y,f(x))d\rho = \int_X \int_Y V(y,f(x))d\rho_x(y)d\rho_X$$

and f_ρ^V is a minimizer of $\mathcal{E}(f)$.

The approximation error measures the approximation ability of the space $\mathcal{H}_K$ with respect to the learning process involving V and ρ. The denseness of $\mathcal{H}_K$ in $C(X)$ implies $\lim_{\lambda\to 0}\mathcal{D}(\lambda) = 0$. A natural assumption would be

$$\mathcal{D}(\lambda) \le \mathcal{D}_0\lambda^\beta \qquad \text{for some } 0 \le \beta \le 1 \text{ and } \mathcal{D}_0 > 0. \tag{12}$$

Throughout the paper we assume for the general loss function V that $\|V\| := \sup_{y\in Y} V(y,0) + \sup\{|V(y,f) - V(y,0)|/|f| : y \in Y, |f| \le 1\} < \infty$.

Remark 13 *Since $\mathcal{D}(\lambda) \le \mathcal{E}(0) + 0 \le \|V\|$ for any $\lambda > 0$, we see that (12) always holds with $\beta = 0$ and $\mathcal{D}_0 = \|V\|$.*

For the least square loss $V(y,f) = (y-f)^2$, the minimizer f_ρ^V of $\mathcal{E}(f)$ is exactly the regression function defined by (5) and approximation error (11) takes the from $\mathcal{D}(\lambda) = \inf_{f\in\mathcal{H}_K}\{\|f - f_\rho\|^2_{L^2_{\rho_X}} + \frac{\lambda}{2}\|f\|_K^2\}$ which measures the approximation of f_ρ in $L^2_{\rho_X}$ by functions from the RKHS $\mathcal{H}_K$.

For a general loss function V, the minimizer f_ρ^V of $\mathcal{E}(f)$ is in general different from f_ρ. Moreover, the approximation error $\mathcal{D}(\lambda)$ involves the approximation of f_ρ^V by $\mathcal{H}_K$ in some function spaces which need not be $L^2_{\rho_X}$.

Example 5 *For the hinge loss $V(y,f) = (1-yf)_+$, the minimizer f_ρ^V is the Bayes rule $f_\rho^V = f_c$ (Devroye et al., 1997). Moreover the uniform Lipschitz continuity of $V(\cdot,f)$ implies (Chen et al., 2004)*

$$\mathcal{E}(f) - \mathcal{E}(f_\rho^V) = \int_Z \phi(yf(x)) - \phi(yf_c(x))d\rho \le \int_X |f(x) - f_c(x)|d\rho_X = \|f - f_c\|_{L^1_{\rho_X}}.$$

So approximation error (11) can be estimated by approximation in the space $L^1_{\rho_X}$ and condition (9) implies (12).

Consider the insensitive loss $V = V_{in}$. We can easily see that for each $x \in X$, $f_\rho^{V_{in}}(x)$ equals the median of the probability distribution ρ_x on Y. That is, $f_\rho^{V_{in}}(x)$ is uniquely determined by

$$\rho_x(\{y \in Y : y \le f_\rho^{V_{in}}(x)\}) \ge \frac{1}{2} \quad \text{and} \quad \rho_x(\{y \in Y : y \ge f_\rho^{V_{in}}(x)\}) \ge \frac{1}{2}.$$

When ρ_x is symmetric about its mean $f_\rho(x)$, we have $f_\rho^{V_{in}}(x) = f_\rho(x)$.

Example 6 *Let $V = V_{in}$, $f_\rho \in C(X)$ and $p \ge 1$. If for each $x \in X$, the conditional distribution ρ_x is given by*

$$d\rho_x(y) = \begin{cases} \frac{p}{2}|y - f_\rho(x)|^{p-1}dy, & \text{if } |y - f_\rho(x)| \le 1, \\ 0, & \text{if } |y - f_\rho(x)| > 1, \end{cases}$$

then we have

$$\begin{aligned} \mathcal{E}(f) - \mathcal{E}(f_\rho^{V_{in}}) &= \frac{1}{p+1}\|f - f_\rho\|^{p+1}_{L^{p+1}_{\rho_X}} - \frac{1}{p+1}\int_{\{x \in X : |f(x) - f_\rho(x)| > 1\}} \\ &\quad |f(x) - f_\rho(x)|^{p+1} + p - (1+p)|f(x) - f_\rho(x)|d\rho_X. \end{aligned}$$

It follows that

$$\mathcal{D}(\lambda) \le \inf_{f \in \mathcal{H}_K} \left\{ \|f - f_\rho\|^{p+1}_{L^{p+1}_{\rho_X}} + \frac{\lambda}{2}\|f\|^2_K \right\}.$$

The conclusion of Example 6 will be proved in the appendix. The following general result follows from the same argument as in Wu et al. (2006).

Proposition 14 *If the loss function V satisfies $|V(y, f_1) - V(y, f_2)| \le |f_1 - f_2|^c$ for some $0 < c \le 1$ and any $y \in Y, f_1, f_2 \in \mathbb{R}$, then*

$$\mathcal{D}(\lambda) \le \inf_{f \in \mathcal{H}_K} \{\|f - f_\rho^V\|^c_{L^1_{\rho_X}} + \frac{\lambda}{2}\|f\|^2_K\} \le \inf_{f \in \mathcal{H}_K} \{\|f - f_\rho^V\|^c_{L^2_{\rho_X}} + \frac{\lambda}{2}\|f\|^2_K\}.$$

If the univariate function $V(y, \cdot)$ is C^1 and satisfies $|\partial V(y, f_1) - \partial V(y, f_2)| \le |f_1 - f_2|^c$ for some $0 < c \le 1$ and any $y \in Y, f_1, f_2 \in \mathbb{R}$, then

$$\mathcal{D}(\lambda) \le \inf_{f \in \mathcal{H}_K} \{\|f - f_\rho^V\|^{1+c}_{L^{1+c}_{\rho_X}} + \frac{\lambda}{2}\|f\|^2_K\} \le \inf_{f \in \mathcal{H}_K} \{\|f - f_\rho^V\|^{1+c}_{L^2_{\rho_X}} + \frac{\lambda}{2}\|f\|^2_K\}.$$

3. Key Analysis for the Fully Online Non-identical Setting

Learning rates for fully online learning algorithm (3) such as those stated in the last section for regression and classification are obtained through analysis for approximation error and sample error. The approximation error has been well understood (Smale and Zhou, 2003; Chen et al., 2004). The sample error for (3) will be estimated in the following two sections. It is expressed as $\|f_{T+1} - f^V_{\lambda_T}\|_K$ where $f^V_{\lambda_T}$ is a regularizing function.

Definition 15 *For $\lambda > 0$ the regularizing function $f_\lambda^V \in \mathcal{H}_K$ is defined by*

$$f_\lambda^V = \arg \inf_{f \in \mathcal{H}_K} \left\{ \mathcal{E}(f) + \frac{\lambda}{2} \|f\|_K^2 \right\}. \tag{13}$$

Observe that f_λ^V is a sample-free limit of $f_{\mathbf{z},\lambda}$ defined by (6). It is natural to expect that the function f_{T+1} produced by online algorithm (3) approximates the regularizing function $f_{\lambda_T}^V$ well. This is actually the case. Our main result on sample error analysis estimates the difference $f_{T+1} - f_{\lambda_T}^V$ in the $\mathcal{H}_K$-norm for quite general situations. The estimation follows from an iteration procedure, developed for online classification algorithms in Ying and Zhou (2006), Ying (2007) and Ye and Zhou (2007), together with some novelty provided in the proof of Theorem 26 in the next section. It is based on a one-step iteration bounding $\|f_{t+1} - f_{\lambda_t}^V\|_K$ in terms of $\|f_t - f_{\lambda_{t-1}}^V\|_K$. The goal of this section is to present our key analysis for one-step iteration in tackling two barriers arising from the fully online non-identical setting.

3.1 Bounding Error Term Caused by Non-identical Sequence of Distributions

The first part of our key analysis for one-step iteration is to tackle an extra error term Δ_t caused by the non-identical sequence of distributions when bounding $\|f_{t+1} - f_{\lambda_t}^V\|_K$ by means of $\|f_t - f_{\lambda_t}^V\|_K$ with the fixed regularization parameter λ_t.

Lemma 16 *Define $\{f_t\}$ by (3). Then we have*

$$\mathbb{E}_{z_t}(\|f_{t+1} - f_{\lambda_t}^V\|_K^2) \le (1 - \eta_t\lambda_t)\|f_t - f_{\lambda_t}^V\|_K^2 + 2\eta_t\Delta_t + \eta_t^2 \mathbb{E}_{z_t} \|\partial V(y_t, f_t(x_t))K_{x_t} + \lambda_t f_t\|_K^2, \tag{14}$$

where Δ_t is defined by

$$\Delta_t = \int_Z \left\{ V(y, f_{\lambda_t}^V(x)) - V(y, f_t(x)) \right\} d\left[\rho^{(t)} - \rho\right]. \tag{15}$$

Lemma 16 follows from the same procedure as in the proof of Lemma 3 in Ying and Zhou (2006) and Ye and Zhou (2007). A crucial estimate is

$$\langle \partial V(y_t, f_t(x_t))K_{x_t} + \lambda_t f_t, f_{\lambda_t}^V - f_t \rangle_K \le [V(y_t, f_{\lambda_t}^V(x_t)) + \frac{\lambda_t}{2}\|f_{\lambda_t}^V\|_K^2] - [V(y_t, f_t(x_t)) + \frac{\lambda_t}{2}\|f_t\|_K^2].$$

When we take the expectation with respect to $z_t = (x_t, y_t)$, we get $\int_Z V(y, f_{\lambda_t}^V(x))d\rho^{(t)}$ (and $\int_Z V(y, f_t(x))d\rho^{(t)}$) on the right-hand side, not $\mathcal{E}(f_{\lambda_t}^V) = \int_Z V(y, f_{\lambda_t}^V(x))d\rho$. So compared with results in the i.i.d. case (Smale and Yao, 2006; Ying and Zhou, 2006; Ying, 2007; Tarrès and Yao, 2005; Ye and Zhou, 2007), an extra term Δ_t involving the difference measure $\rho^{(t)} - \rho$ appears. This is the first barrier we need to tackle here.

When V is the least square loss, it can be easily handled by assuming $f_\rho \in C^s(X)$. In fact, from $V(y, f) = (y - f)^2$, we see that

$$\begin{aligned} \Delta_t &= \int_X \left\{ \int_Y [f_{\lambda_t}^V(x)) - f_t(x)][f_{\lambda_t}^V(x) + f_t(x) - 2y] d\rho_x(y) \right\} d\left[\rho_X^{(t)} - \rho_X\right] \\ &= \int_X [f_{\lambda_t}^V(x)) - f_t(x)][f_{\lambda_t}^V(x) + f_t(x) - 2f_\rho(x)] d\left[\rho_X^{(t)} - \rho_X\right]. \end{aligned}$$

This together with the relation $\|hg\|_{C^s(X)} \le \|h\|_{C^s(X)}\|g\|_{C^s(X)}$ yields the following bound.

Proposition 17 *Let $V(y,f) = (y-f)^2$ and $f_\rho \in C^s(X)$. Then we have*

$$\Delta_t \le \left\{ \|f_{\lambda_t}^V\|_{C^s(X)} + \|f_t\|_{C^s(X)} + 2\|f_\rho\|_{C^s(X)} \right\} \|\rho_X^{(t)} - \rho_X\|_{(C^s(X))^*} \|f_{\lambda_t}^V - f_t\|_{C^s(X)}.$$

For a general loss function and output space, Δ_t can be bounded under assumption (4). It is a special case of the following general result when $h = f_{\lambda_t}^V$ and $g = f_t$.

Lemma 18 *Let $h, g \in C^s(X)$. If (4) holds, then we have*

$$\begin{aligned} &\left| \int_Z V(y,h(x)) - V(y,g(x)) d\left[\rho^{(t)} - \rho\right] \right| \\ &\quad \le \left\{ B_{h,g} \left(\|h\|_{C^s(X)} + \|g\|_{C^s(X)} \right) + 2C_\rho \widetilde{B}_{h,g} \right\} \|\rho_X^{(t)} - \rho_X\|_{(C^s(X))^*}, \end{aligned}$$

where $B_{h,g}$ and $\widetilde{B}_{h,g}$ are constants given by

$$B_{h,g} = \sup\left\{ |\partial V(y,f)| : y \in Y, |f| \le \max\{\|h\|_{C(X)}, \|g\|_{C(X)}\} \right\}$$

and

$$\widetilde{B}_{h,g} = \sup\left\{ \|V(\cdot,f)\|_{C^s(Y)} : |f| \le \max\{\|h\|_{C(X)}, \|g\|_{C(X)}\} \right\}.$$

Proof By decomposing the probability distributions on Z into marginal and conditional distributions, we see

$$\int_Z V(y,h(x)) - V(y,g(x)) d\left[\rho^{(t)} - \rho\right] = \int_X \int_Y V(y,h(x)) - V(y,g(x)) d\rho_x(y) d\left[\rho_X^{(t)} - \rho_X\right].$$

By the definition of the norm in $(C^s(X))^*$, we obtain

$$\left| \int_Z V(y,h(x)) - V(y,g(x)) d\left[\rho^{(t)} - \rho\right] \right| \le \|\rho_X^{(t)} - \rho_X\|_{(C^s(X))^*} \|J\|_{C^s(X)},$$

where J is a function on X defined by

$$J(x) = \int_Y V(y,h(x)) - V(y,g(x)) d\rho_x(y), \qquad x \in X.$$

The notion of $B_{h,g}$ tells us that $|V(y,h(x)) - V(y,g(x))| \le B_{h,g}|h(x) - g(x)|$ for each $y \in Y$. Hence $\|J\|_{C(X)} \le B_{h,g}\|h - g\|_{C(X)}$.

To bound $|J|_{C^s(X)}$, let $x, u \in X$. We can decompose $J(x) - J(u)$ as

$$\begin{aligned} J(x) - J(u) &= \int_Y \{[V(y,h(x)) - V(y,g(x))] - [V(y,h(u)) - V(y,g(u))]\} d\rho_x(y) \\ &\quad + \int_Y [V(y,h(u)) - V(y,g(u))] d[\rho_x - \rho_u](y). \end{aligned}$$

By the notion $B_{h,g}$, part of the first term above can be bounded as

$$|V(y,h(x)) - V(y,h(u))| \le B_{h,g}|h(x) - h(u)| \le B_{h,g}|h|_{C^s(X)}(d(x,u))^s.$$

The same bound holds for the other part of the first term. So we get

$$\left|\int_Y \{[V(y,h(x))-V(y,g(x))]-[V(y,h(u))-V(y,g(u))]\}\,d\rho_x(y)\right| \le B_{h,g}\left\{|h|_{C^s(X)}+|g|_{C^s(X)}\right\}(d(x,u))^s.$$

For the other term of the expression for $J(x)-J(u)$, we apply condition (4) to

$$\left|\int_Y [V(y,h(u))-V(y,g(u))]d[\rho_x-\rho_u](y)\right| \le \|\rho_x-\rho_u\|_{(C^s(Y))^*}\|V(y,h(u))-V(y,g(u))\|_{C^s(Y)}$$

and find that

$$\left|\int_Y [V(y,h(u))-V(y,g(u))]d[\rho_x-\rho_u](y)\right| \le C_\rho (d(x,u))^s 2\widetilde{B}_{h,g}.$$

Combining the above two bounds, we see that

$$|J|_{C^s(X)} = \sup_{x\neq u\in X}\frac{|J(x)-J(u)|}{(d(x,u))^s} \le B_{h,g}\left\{|h|_{C^s(X)}+|g|_{C^s(X)}\right\}+2C_\rho\widetilde{B}_{h,g}.$$

Then the desired bound follows and the lemma is proved. ■

3.2 Bounding Error Term Caused by Varying Regularization Parameters

The second part of our key analysis is to estimate the error term called drift error $\|f^V_{\lambda_t}-f^V_{\lambda_{t-1}}\|_K$ caused by the change of the regularization parameter from λ_{t-1} to λ_t in our fully online algorithm. This is the second barrier we need to tackle here.

Definition 19 *The drift error is defined as*

$$d_t = \|f^V_{\lambda_t}-f^V_{\lambda_{t-1}}\|_K.$$

The drift error can be estimated by the approximation error, which has been studied for regression (Smale and Zhou, 2009) and for classification (Ye and Zhou, 2007).

Theorem 20 *Let V be a convex loss function, f^V_λ by (13) and $\mu>\lambda>0$. We have*

$$\|f^V_\lambda-f^V_\mu\|_K \le \frac{\mu}{2}\left(\frac{1}{\lambda}-\frac{1}{\mu}\right)\left(\|f^V_\lambda\|_K+\|f^V_\mu\|_K\right) \le \frac{\mu}{2}\left(\frac{1}{\lambda}-\frac{1}{\mu}\right)\left(\sqrt{\frac{2\mathcal{D}(\lambda)}{\lambda}}+\sqrt{\frac{2\mathcal{D}(\mu)}{\mu}}\right).$$

In particular, if with some $0<\gamma\le 1$ we take $\lambda_t=\lambda_1 t^{-\gamma}$ for $t\ge 1$, then

$$d_{t+1} \le 2t^{\frac{\gamma}{2}-1}\sqrt{\mathcal{D}(\lambda_1 t^{-\gamma})/\lambda_1}.$$

Proof Taking derivative of the functional $\mathcal{E}(f)+\frac{\lambda}{2}\|f\|^2_K$ at the minimizer f^ϕ_λ defined in (13), we see that

$$\int_Z \partial V(y,f^V_\lambda(x))K_x d\rho+\lambda f^V_\lambda = 0.$$

It follows that

$$f_\lambda^V - f_\mu^V = \frac{1}{\mu}\int_Z \partial V(y, f_\mu^V(x))K_x d\rho - \frac{1}{\lambda}\int_Z \partial V(y, f_\lambda^V(x))K_x d\rho.$$

Combining with the reproducing property (8), we know $\|f_\lambda^V - f_\mu^V\|_K^2 = \langle f_\lambda^V - f_\mu^V, f_\lambda^V - f_\mu^V\rangle_K$ can be expressed as

$$\|f_\lambda^V - f_\mu^V\|_K^2 = \frac{1}{\mu}\int_Z \partial V(y, f_\mu^V(x))\left(f_\lambda^V - f_\mu^V\right)(x)d\rho - \frac{1}{\lambda}\int_Z \partial V(y, f_\lambda^V(x))\left(f_\lambda^V - f_\mu^V\right)(x)d\rho.$$

The convexity of the function $V(y,\cdot)$ on $\mathbb{R}$ tells us that

$$\partial V(y, f_\mu^V(x))\left(f_\lambda^V - f_\mu^V\right)(x) \le V(y, f_\lambda^V(x)) - V(y, f_\mu^V(x))$$

and

$$\partial V(y, f_\lambda^V(x))\left(f_\mu^V - f_\lambda^V\right)(x) \le V(y, f_\mu^V(x)) - V(y, f_\lambda^V(x)).$$

Hence

$$\|f_\lambda^V - f_\mu^V\|_K^2 \le (\frac{1}{\lambda} - \frac{1}{\mu})(\mathcal{E}(f_\mu^V) - \mathcal{E}(f_\lambda^V)).$$

¿From the definition of f_μ^V, we see that $\mathcal{E}(f_\mu^V) + \frac{\mu}{2}\|f_\mu^V\|_K^2 - (\mathcal{E}(f_\lambda^V) + \frac{\mu}{2}\|f_\lambda^V\|_K^2) \le 0$. It follows that

$$\mathcal{E}(f_\mu^V) - \mathcal{E}(f_\lambda^V) \le \frac{\mu}{2}(\|f_\lambda^V\|_K^2 - \|f_\mu^V\|_K^2) \le \frac{\mu}{2}\|f_\lambda^V - f_\mu^V\|_K(\|f_\lambda^V\|_K + \|f_\mu^V\|_K).$$

Then the desired inequality follows. ■

4. Bounds for Sample Error in $\mathcal{H}_K$-norm

We are in a position to present our main result on the sample error measured with the $\mathcal{H}_K$-norm of the difference $f_{T+1} - f_{\lambda_T}^V$. This will be done by applying iteratively the key analysis in Lemma 16, Lemma 18 and Theorem 20.

When applying Lemma 18, we need to bound $\|g\|_{C^s(X)}$ in terms of $\|g\|_K$.

Definition 21 *We say that the Mercer kernel K satisfies the kernel condition of order s if $K \in C^s(X\times X)$ and for some $\kappa_{2s} > 0$,*

$$|K(x,x) - 2K(x,u) + K(u,u)| \le \kappa_{2s}^2 (d(x,u))^{2s}, \qquad \forall x,u \in X. \tag{16}$$

When $0 < s \le \frac{1}{2}$ and $K \in C^{2s}(X\times X)$, (16) holds true. The following result follows directly from Zhou (2003), Zhou (2008) and Smale and Zhou (2009).

Lemma 22 *If K satisfies the kernel condition of order s with (16) valid, then we have*

$$\|g\|_{C^s(X)} \le (\kappa + \kappa_{2s})\|g\|_K, \qquad \forall g \in \mathcal{H}_K.$$

When using Lemma 16 and Lemma 18, we need incremental behaviors of the loss function V to bound $\|f_t\|_K$.

Definition 23 *Denote*

$$N(\lambda) = \sup\left\{|\partial V(y,f)| : y \in Y, |f| \le \max\left\{\kappa^2\|V\|/\lambda, \kappa\sqrt{2\|V\|/\lambda}\right\}\right\}$$

and

$$\widetilde{N}(\lambda) = \sup\left\{\|V(\cdot,f)\|_{C^s(Y)} : y \in Y, |f| \le \max\left\{\kappa^2\|V\|/\lambda, \kappa\sqrt{2\|V\|/\lambda}\right\}\right\}.$$

We say that V has incremental exponent $p \ge 0$ if for some $N_1 > 0$ and $\lambda_1 > 0$ we have

$$N(\lambda) \le N_1(\frac{1}{\lambda})^p \quad \text{and} \quad \widetilde{N}(\lambda) \le N_1(\frac{1}{\lambda})^{p+1} \qquad \forall 0 < \lambda \le \lambda_1. \tag{17}$$

We say that ∂V is locally Lipschitz at the origin if

$$M_0 := \sup\left\{\frac{|\partial V(y,f) - \partial V(y,0)|}{|f|} : y \in Y, |f| \le 1\right\} < \infty. \tag{18}$$

The following result can be proved by exactly the same procedure as those in Ying and Zhou (2006), Ying (2007) and Ye and Zhou (2007).

Lemma 24 *Assume that ∂V is locally Lipschitz at the origin. Define $\{f_t\}$ by (3). If*

$$\eta_t\left(\kappa^2(M_0 + 2N(\lambda_t)) + \lambda_t\right) \le 1 \tag{19}$$

for $t = 1, \ldots, T$, then we have

$$\|f_t\|_K \le \frac{\kappa\|V\|}{\lambda_t}, \qquad t = 1, \ldots, T+1. \tag{20}$$

For the insensitive loss, (18) is not satisfied, but $\partial V(y,f)$ is uniformly bounded by 1. For such loss functions we can apply the following bound.

Lemma 25 *Assume $\|\partial V(y,f)\|_\infty := \sup_{y\in Y, f\in\mathbb{R}} |\partial V(y,f)| < \infty$. Define $\{f_t\}$ by (3). If for some $\lambda_1, \eta_1 > 0$ and $\gamma, \alpha > 0$ with $\lambda + \alpha < 1$, we take $\lambda_t = \lambda_1 t^{-\gamma}, \eta_t = \eta_1 t^{-\alpha}$ with $t = 1, \ldots, T$, then we have*

$$\|f_t\|_K \le \frac{C_{V,\gamma,\alpha}}{\lambda_t}, \qquad t = 1, \ldots, T+1, \tag{21}$$

where $C_{V,\gamma,\alpha}$ is the constant given by

$$C_{V,\gamma,\alpha} = \kappa\|\partial V(y,f)\|_\infty\left\{\lambda_1\eta_1 + \lambda_1\left(2^{\gamma+2\alpha}/(\lambda_1\eta_1) + \left((1+\alpha)/[e\lambda_1\eta_1(1-2^{\gamma+\alpha-1})]\right)^{\frac{1+\alpha}{1-\gamma-\alpha}}\right)\right\}.$$

Proof By taking norms in (3) we see that

$$\|f_{t+1}\|_K \le (1 - \eta_t\lambda_t)\|f_t\|_K + \eta_t\kappa\|\partial V(y,f)\|_\infty.$$

By iterating and the choice $f_1 = 0$ we find

$$\|f_{t+1}\|_K \le \sum_{i=1}^{t} \Pi_{j=i+1}^{t}(1 - \eta_j\lambda_j)\eta_i\kappa\|\partial V(y,f)\|_\infty.$$

But $1-\eta_j\lambda_j \le \exp\{-\eta_j\lambda_j\}$. It follows that

$$\|f_{t+1}\|_K \le \sum_{i=1}^{t} \Pi_{j=i+1}^{t} \exp\left\{-\sum_{j=i+1}^{t}\eta_j\lambda_j\right\}\eta_i\kappa\|\partial V(y,f)\|_\infty.$$

Now we need the following elementary inequality with $c>0, q_2\ge 0$ and $0<q_1<1$:

$$\sum_{i=1}^{t-1} i^{-q_2}\exp\left\{-c\sum_{j=i+1}^{t} j^{-q_1}\right\} \le \left(\frac{2^{q_1+q_2}}{c}+\left(\frac{1+q_2}{ec(1-2^{q_1-1})}\right)^{\frac{1+q_2}{1-q_1}}\right)t^{q_1-q_2}. \tag{22}$$

This elementary inequality can be found in, for example, Smale and Zhou (2009). Taking $q_2=\alpha, q_1=\gamma+\alpha$ and $c=\lambda_1\eta_1$ we know that $\|f_{t+1}\|_K$ is bounded by

$$\kappa\|\partial V(y,f)\|_\infty\left\{\eta_1 t^{-\alpha}+\left(2^{\gamma+2\alpha}/(\lambda_1\eta_1)+\left((1+\alpha)/[e\lambda_1\eta_1(1-2^{\gamma+\alpha-1})]\right)^{\frac{1+\alpha}{1-\gamma-\alpha}}\right)t^{\gamma}\right\}.$$

Then our desired bound holds true. ■

Now we can present our bound for the sample error $\|f_{T+1}-f_{\lambda_T}^V\|_K$.

Theorem 26 *Suppose the following assumptions hold:*

1. *the kernel K satisfies the kernel condition of order s ($0<s\le 1$) with (16) valid.*
2. *V has incremental exponent $p\ge 0$ with (17) valid and ∂V satisfies (18).*
3. *$\{\rho_X^{(t)}\}_{t=1,2,\cdots}$ converges polynomially to ρ_X in $(C^s(X))^*$ with (1) valid.*
4. *the distributions $\{\rho_x : x\in X\}$ is Lipschitz s in $(C^s(Y))^*$ with (4) valid.*
5. *the triple (K,V,ρ) has the approximation ability of power $0<\beta\le 1$ stated by (12).*

Take

$$\lambda_t=\lambda_1 t^{-\gamma}, \eta_t=\eta_1 t^{-\alpha} \tag{23}$$

with some $\lambda_1,\eta_1>0$ and γ,α satisfying

$$0<\gamma<\frac{2}{5+4p-\beta},\ (2p+1)\gamma<\alpha<1-\frac{\gamma(3-\beta)}{2},\ \eta_1\le\frac{1}{\kappa^2M_0+2\kappa^2N_1\lambda_1^{-p}+\lambda_1}. \tag{24}$$

Then we have

$$\mathbb{E}_{z_1,z_2,\cdots,z_T}(\|f_{T+1}-f_{\lambda_T}^V\|_K^2)\le C_{K,V,\rho,b,\beta,s}T^{-\theta} \tag{25}$$

where the power index θ is given by

$$\theta:=\min\left\{2-\gamma(3-\beta)-2\alpha,\alpha-\gamma(2p+1),b-\gamma(2+p)\right\}, \tag{26}$$

and $C_{K,V,\rho,b,\beta,s}$ is a constant independent of T given explicitly in the proof.

Proof We divide the proof into four steps.

First we bound Δ_t. Since λ_t and η_t take the form (23), we see from the lower bound for α in (24) that $p\gamma \le (2p+1)\gamma \le \alpha$. Hence for $t \in \mathbb{N}$,

$$\eta_t(\kappa^2(M_0+2N(\lambda_t))+\lambda_t) \le \eta_1 t^{-\alpha}(\kappa^2 M_0 + 2\kappa^2 N_1 \lambda_1^{-p} t^{p\gamma} + \lambda_1 t^{-\gamma}) \le \eta_1(\kappa^2 M_0 + 2\kappa^2 N_1 \lambda_1^{-p} + \lambda_1).$$

So the last restriction of (24) implies (19), and by Lemma 24, we know that (20) holds true.

Taking $f = 0$ in (13) yields

$$\|f_{\lambda_t}^V\|_K^2 \le \frac{2\|V\|}{\lambda_t}, \qquad t = 1, \ldots, T.$$

Putting these bounds into the definition of constant $B_{h,g}, \widetilde{B}_{h,g}$, we see by the notion $N(\lambda)$ that

$$B_{f_{\lambda_t}^V, f_t} \le N(\lambda_t), \qquad \widetilde{B}_{f_{\lambda_t}^V, f_t} \le N(\lambda_t).$$

So by Lemma 18 and Lemma 22,

$$\Delta_t \le B_t^* := \left\{(\kappa+\kappa_{2s})(\sqrt{2\|V\|/\lambda_t}+\kappa\|V\|/\lambda_t)N(\lambda_t) + 2C_\rho \widetilde{N}(\lambda_t)\right\} \|\rho_X^{(t)} - \rho_X\|_{(C^s(X))^*}.$$

Putting this bound and (20) into (14) yields

$$\mathbb{E}_{z_1,z_2,\cdots,z_t}(\|f_{t+1} - f_{\lambda_t}^V\|_K^2) \le (1-\eta_t\lambda_t)\mathbb{E}_{z_1,z_2,\cdots,z_{t-1}}\left(\|f_t - f_{\lambda_t}^V\|_K^2\right) + \kappa^2\eta_t^2(N(\lambda_t)+\|V\|)^2 + 2\eta_t B_t^*.$$

Next we derive explicit bounds for the one-step iteration.

Recall that $d_t = \|f_{\lambda_t}^V - f_{\lambda_{t-1}}^V\|_K$. It gives $\|f_t - f_{\lambda_t}^V\|_K^2 \le \|f_t - f_{\lambda_{t-1}}^V\|_K^2 + 2\|f_t - f_{\lambda_{t-1}}^V\|_K d_t + d_t^2$.

Take $\tau = \frac{\gamma+\alpha}{1-\gamma(1-\beta)/2}$. By the upper bound for α in (24), we find $0 < \tau < 1$.

Take $A_1 = \frac{\eta_1 \lambda_1^{1+\tau(1-\beta)/2}}{2^{1+2\tau}\mathcal{D}_0^{\tau/2}} > 0$. Applying the elementary inequality

$$2ab = 2[\sqrt{A_1}ab^{\tau/2}][b^{1-\tau/2}/\sqrt{A_1}] \le A_1 a^2 b^\tau + b^{2-\tau}/A_1 \tag{27}$$

to $a = \|f_t - f_{\lambda_{t-1}}^V\|_K$ and $b = d_t$, we know that

$$\mathbb{E}_{z_1,z_2,\cdots,z_{t-1}}\left(\|f_t - f_{\lambda_t}^V\|_K^2\right) \le (1+A_1 d_t^\tau)\mathbb{E}_{z_1,z_2,\cdots,z_{t-1}}\left(\|f_t - f_{\lambda_{t-1}}^V\|_K^2\right) + d_t^{2-\tau}/A_1 + d_t^2.$$

Using this bound and noticing the inequality $(1-\eta_t\lambda_t)(1+A_1 d_t^\tau) \le 1 + A_1 d_t^\tau - \eta_t\lambda_t$, we obtain

$$\begin{aligned}\mathbb{E}_{z_1,z_2,\cdots,z_t}(\|f_{t+1} - f_{\lambda_t}^V\|_K^2) \le (1+A_1 d_t^\tau - \eta_t\lambda_t)\mathbb{E}_{z_1,z_2,\cdots,z_{t-1}}\left(\|f_t - f_{\lambda_{t-1}}^V\|_K^2\right) \\ + d_t^{2-\tau}/A_1 + d_t^2 + \kappa^2\eta_t^2(N(\lambda_t)+\|V\|)^2 + 2\eta_t B_t^*.\end{aligned}$$

By Theorem 20, condition (12) for the approximation error yields

$$d_t \le 2\sqrt{\mathcal{D}_0}\lambda_1^{(\beta-1)/2}(t-1)^{\gamma(1-\beta)/2-1} \le A_2 t^{\gamma(1-\beta)/2-1} \quad \text{where } A_2 = 4\sqrt{\mathcal{D}_0}\lambda_1^{(\beta-1)/2}.$$

Inserting the parameter form $\lambda_t = \lambda_1 t^{-\gamma}$ into assumption (17) and applying condition (1), we can bound B_t^* as

$$B_t^* \le A_3 t^{\gamma(1+p)-b} \quad \text{where } A_3 = \left\{(\kappa+\kappa_{2s})(\sqrt{2\|V\|/\lambda_1}+\kappa\|V\|/\lambda_1) + 2C_\rho/\lambda_1\right\} N_1 \lambda_1^{-p} C.$$

Therefore, for the one-step iteration, by denoting $f_{\lambda_1}^V$ as $f_{\lambda_0}^V$ when $t=1$, we have for each $t=1,\ldots,T$,

$$\mathbb{E}_{z_1,z_2,\cdots,z_t}(\|f_{t+1}-f_{\lambda_t}^V\|_K^2) \le (1+A_1 d_t^{\tau}-\eta_t\lambda_t)\mathbb{E}_{z_1,z_2,\cdots,z_{t-1}}\left(\|f_t-f_{\lambda_{t-1}}^V\|_K^2\right)+A_4 t^{-\widetilde{\theta}}, \tag{28}$$

where

$$\widetilde{\theta}=\min\left\{2-\gamma(2-\beta)-\alpha,\ 2(\alpha-p\gamma),\ \alpha+b-\gamma(1+p)\right\}$$

and

$$A_4=A_2^{2-\tau}/A_1+A_2^2+\kappa^2\eta_1^2(N_1\lambda_1^{-p}+\|V\|)^2+2\eta_1 A_3.$$

Then we iterate the above one-step analysis. Inserting the parameter forms for λ_t and η_t, we see from the definition of the constant A_1 that

$$1+A_1 d_t^{\tau}-\eta_t\lambda_t \le 1+A_1A_2^{\tau}t^{\tau(\gamma(1-\beta)/2-1)}-\eta_1\lambda_1 t^{-\gamma-\alpha}=1-\frac{\eta_1\lambda_1}{2}t^{-\gamma-\alpha}. \tag{29}$$

So the one-step analysis (28) yields

$$\mathbb{E}_{z_1,z_2,\cdots,z_t}(\|f_{t+1}-f_{\lambda_t}^V\|_K^2) \le \left(1-\frac{\eta_1\lambda_1}{2}t^{-\gamma-\alpha}\right)\mathbb{E}_{z_1,z_2,\cdots,z_{t-1}}\left(\|f_t-f_{\lambda_{t-1}}^V\|_K^2\right)+A_4t^{-\widetilde{\theta}}.$$

Applying this bound iteratively for $t=1,\ldots,T$ implies

$$\begin{aligned}\mathbb{E}_{z_1,z_2,\cdots,z_T}(\|f_{T+1}-f_{\lambda_T}^V\|_K^2) &\le A_4\sum_{t=1}^{T}\Pi_{j=t+1}^{T}(1-\frac{\eta_1\lambda_1}{2}j^{-\gamma-\alpha})t^{-\widetilde{\theta}}\\ &+\left\{\Pi_{t=1}^{T}(1-\frac{\eta_1\lambda_1}{2}t^{-\gamma-\alpha})\right\}\|f_1-f_{\lambda_1}^V\|_K^2.\end{aligned}$$

Finally we bound the above expressions by two elementary inequalities. The first one is (22). Applying this inequality with $c=\frac{\eta_1\lambda_1}{2}, q_1=\gamma+\alpha$ and $q_2=\widetilde{\theta}$, since $1-u\le e^{-u}$ for any $u\ge 0$, the first expression above can be bounded as

$$\sum_{t=1}^{T}\Pi_{j=t+1}^{T}(1-\frac{\eta_1\lambda_1}{2}j^{-\gamma-\alpha})t^{-\widetilde{\theta}} \le \sum_{t=1}^{T}\exp\left\{-\frac{\eta_1\lambda_1}{2}\sum_{j=t+1}^{T}j^{-\gamma-\alpha}\right\}t^{-\widetilde{\theta}} \le A_5T^{\gamma+\alpha-\widetilde{\theta}},$$

where A_5 is the constant given by

$$A_5=\frac{2^{\gamma+\alpha+\widetilde{\theta}+1}}{\eta_1\lambda_1}+1+\left(\frac{2+2\widetilde{\theta}}{e\eta_1\lambda_1(1-2^{\gamma+\alpha-1})}\right)^{\frac{1+\widetilde{\theta}}{1-\gamma-\alpha}}.$$

For the second expression above, we have

$$\begin{aligned}&\Pi_{t=1}^{T}(1-\frac{\eta_1\lambda_1}{2}t^{-\gamma-\alpha}) \le \exp\left\{-\frac{\eta_1\lambda_1}{2}\sum_{j=1}^{T}t^{-\gamma-\alpha}\right\} \le \exp\left\{-\frac{\eta_1\lambda_1}{2}\int_1^{T+1}x^{-\gamma-\alpha}dx\right\}\\ &\le \exp\left\{\frac{\lambda_1\eta_1}{2(1-\gamma-\alpha)}\right\}\exp\left\{-\frac{\lambda_1\eta_1}{2(1-\gamma-\alpha)}(T+1)^{1-\gamma-\alpha}\right\}.\end{aligned}$$

Applying another elementary inequality

$$\exp\{-cx\} \le \left(\frac{a}{ec}\right)^a x^{-a}, \qquad \forall c,a,x>0$$

with $c=\frac{\lambda_1\eta_1}{2(1-\gamma-\alpha)}$, $a=\frac{2}{1-\gamma-\alpha}$ and $x=(T+1)^{1-\gamma-\alpha}$ yields

$$\Pi_{t=1}^T(1-\frac{\eta_1\lambda_1}{2}t^{-\gamma-\alpha}) \le \exp\left\{\frac{\lambda_1\eta_1}{2(1-\gamma-\alpha)}\right\}\left(\frac{4}{e\lambda_1\eta_1}\right)^{\frac{2}{1-\gamma-\alpha}} T^{-2}.$$

The above two estimates give the desired bound (25) with $\theta=\widetilde{\theta}-\gamma-\alpha$ and the constant $C_{K,V,\rho,b,\beta,s}$ given by

$$C_{K,V,\rho,b,\beta,s} = A_4A_5 + \exp\left\{\frac{\lambda_1\eta_1}{2(1-\gamma-\alpha)}\right\}\left(\frac{4}{e\lambda_1\eta_1}\right)^{\frac{2}{1-\gamma-\alpha}}\frac{2\|V\|}{\lambda_1}.$$

This proves the theorem. ■

Remark 27 *Some ideas in the above proof are from Ying and Zhou (2006), Ye and Zhou (2007) and Smale and Zhou (2009). Two novel points are presented for the first time here. One is the bound for Δ_t, dealing with $\rho_X^{(t)}-\rho_X$, given in the first step of our proof in order to tackle the technical difficulty arising from the non-identical sampling process. The same difficulty for the least square regression was overcome in Smale and Zhou (2009) by the special linear feature and explicit expressions offered by the least square loss. The second technical novelty is to introduce a parameter A_1 into elementary inequality (27). With this parameter, we can bound $1+A_1d_t^\tau-\eta_t\lambda_t$ by $1-\frac{1}{2}\eta_t\lambda_t$, shown in (29). This improves the error bound even in the i.i.d. case presented in Ye and Zhou (2007) for the fully online algorithm.*

Let us discuss the role of parameters in Theorem 26. When γ is small enough and $b>\frac{2}{3}$, fully online algorithm (3) becomes very close to the partially online scheme with $\lambda_t\equiv\lambda_1$. By taking $\alpha=\frac{2}{3}$, the rates in (25) are of order $O(T^{-(\frac{2}{3}-\varepsilon)})$ with ε arbitrarily small, which is a nice bound for the sample error $\|f_{T+1}-f_{\lambda_T}^V\|_K$. In this case, the difference between $f_{\lambda_T}^V$ and f_ρ^V, measured by the approximation error, increases since $\lambda_T=\lambda_1T^{-\gamma}$. To estimate the total error between f_{T+1} and f_ρ^V, we should take a balance for the index γ of the regularization parameter, as shown in Theorem 11.

For the insensitive loss, (18) is not satisfied. We can apply Lemma 25 and obtain bounds for $\|f_{T+1}-f_{\lambda_T}^V\|_K$ by the same proof as that for Theorem 26.

Proposition 28 *Assume $\|\partial V(y,f)\|_\infty<\infty$ and all the conditions of Theorem 26 except (18). Take $\{\lambda_t,\eta_t\}$ by (23) with the restriction (24) without the last inequality. Then the same convergence rate (25) holds true with the power index θ given by (26).*

5. Bounds for Binary Classification and Regression with Insensitive Loss

We demonstrate how to apply Theorem 26 by deriving learning rates of fully online algorithm (3) for binary classification and regression with insensitive loss.

Theorem 29 *Let $V(y,f)=\phi(yf)$ where $\phi:\mathbb{R}\to\mathbb{R}_+$ is a convex function satisfying*

$$|\phi'_-(u)|\le N_\phi|u|^p,\quad \phi(u)\le N_\phi u^{p+1}\qquad \forall |u|\ge 1 \tag{30}$$

for some $p\ge 0$ and $N_\phi>0$. Suppose $M_\phi:=\sup\{|\phi'_-(u)-\phi'_-(0)|/|u| : |u|\le 1\}<\infty$. Assume (16) for K, (1) for $\{\rho_X^{(t)}\}$, and (12) for (K,ϕ,ρ). If $f_\rho\in C^s(X)$ and we choose $\{\lambda_t,\eta_t\}$ as (23) with $0<\gamma<\frac{2}{5+10p-\beta},\alpha=\frac{2+\gamma(2p-2+\beta)}{3}$ and $b>\gamma(2+p)$, and $\eta_1<\eta_0$, $\lambda_1\le\kappa^2(\phi(0)+\|\phi'_-\|_{L^\infty[-1,1]})/2$, then we have

$$\mathbb{E}_{z_1,\dots,z_T}\left(\mathcal{E}(f_{T+1})-\mathcal{E}(f_\rho^V)\right)\le \widetilde{C}_{\phi,\beta,\gamma}T^{-\min\left\{\frac{2-\gamma(5+10p-\beta)}{6},\frac{b-\gamma(2+3p)}{2},\gamma\beta\right\}},$$

where $\eta_0:=\frac{1}{\kappa^2M_\phi+2\kappa^2N_1\lambda_1^{-p}+\lambda_1}$ with $N_1=(M_\phi+N_\phi+\phi(0)+\|\phi'_-\|_{L^\infty[-1,1]})\kappa^{2p}(\phi(0)+\|\phi'_-\|_{L^\infty[-1,1]})^p$ and $\widetilde{C}_{\phi,\beta,\gamma}$ is a constant depending on $\eta_1,\lambda_1,\kappa,\mathcal{D}_0,\beta,\phi,\beta$ and s.

Proof By the bounds for $\|f_{\lambda_T}^V\|_K$ and $\|f_{T+1}\|_K$, we know from (30) that

$$\begin{aligned}\left|\mathcal{E}(f_{T+1})-\mathcal{E}(f_{\lambda_T}^V)\right| &= \left|\int_Z\phi(yf_{T+1}(x))-\phi(yf_{\lambda_T}^V(x))d\rho\right|\\ &\le C_{K,\phi}\lambda_T^{-p}\|f_{T+1}-f_{\lambda_T}^V\|_\infty\le\kappa C_{K,\phi}\lambda_T^{-p}\|f_{T+1}-f_{\lambda_T}^V\|_K,\end{aligned}$$

where $C_{K,\phi}$ is a constant depending on K and ϕ.

It is easy to check that the loss $V(y,f)=\phi(yf)$ satisfies (17) with incremental exponent p.

By Theorem 26 with $0<\gamma<\frac{2}{5+10p-\beta},\alpha=\frac{2+\gamma(2p-2+\beta)}{3}$ and $b>\gamma(2+p)$, we have

$$\mathbb{E}_{z_1,z_2,\cdots,z_T}\left(\left\|f_{T+1}-f_{\lambda_T}^V\right\|_K\right)\le\sqrt{C_{K,V,\rho,b,\beta,s}}T^{-\min\{[2-\gamma(5+4p-\beta)]/6,[b-\gamma(2+p)]/2\}}.$$

Also, we have $\mathcal{E}(f_{\lambda_T}^\phi)-\mathcal{E}(f_\rho^V)\le\mathcal{D}(\lambda_T)\le\mathcal{D}_0\lambda_T^\beta$. Thus we get a bound for the excess generalization error

$$\mathbb{E}_{z_1,\dots,z_T}(\mathcal{E}(f_{T+1})-\mathcal{E}(f_\rho^V))\le\widetilde{C}_{\phi,\beta,\gamma}T^{-\min\{[2-\gamma(5+10p-\beta)]/6,\gamma\beta,[b-\gamma(2+3p)]/2\}},$$

where $\widetilde{C}_{\phi,\beta,\gamma}=\kappa C_{K,\phi}\lambda_1^{-p}\sqrt{C_{K,V,\rho,b,\beta,s}}+\mathcal{D}_0\lambda_1^\beta$. This verifies the desired bound. ■

Theorem 29 yields concrete learning rates with various loss functions. When ϕ is chosen to be the hinge loss $\phi(x)=(1-x)_+$, we can prove Theorem 11.

5.1 Proof of Theorem 11

When $0<s\le\frac{1}{2}$ and $K\in C^{2s}(X\times X)$, (16) holds true.

The loss function $\phi(x)=(1-x)_+$ satisfies $\phi'_-(x)=-1$ for $x\le 1$ and 0 otherwise. It follows that (17) holds true with $p=0$ and $M_\phi=0$. By Example 5, (9) implies (12).

Thus all conditions in Theorem 29 are satisfied and by taking $p=0$ and $\gamma=\frac{1}{4}$, we have

$$\mathbb{E}_{z_1,\dots,z_T}\left(\mathcal{E}(f_{T+1})-\mathcal{E}(f_\rho^V)\right)\le\widetilde{C}_{\phi,\beta,\gamma}T^{-\min\left\{\frac{1}{8}+\frac{\beta}{24},\frac{\beta}{4},\frac{b}{2}-\frac{1}{4}\right\}}.$$

An important relation concerning the hinge loss is the one (Zhang, 2004) between the excess misclassification error and the excess generalization error given for any measurable function $f : X \to \mathbb{R}$ as

$$\mathcal{R}(\mathrm{sgn}(f)) - \mathcal{R}(f_c) \le \mathcal{E}(f) - \mathcal{E}(f_c).$$

Combining this relation with the above bound for the excess generalization error proves the conclusion of Theorem 11. ■

Turn to the general loss ϕ. We give an additional assumption that $\phi''(0)$ exists and is positive. Under this assumption it was proved in Chen et al. (2004) and Bartlett et al. (2006) that there exists a constant depending c_ϕ only on ϕ such that for any measurable function $f : X \to \mathbb{R}$,

$$\mathcal{R}(\mathrm{sgn}(f)) - \mathcal{R}(f_c) \le c_\phi \sqrt{\mathcal{E}(f) - \mathcal{E}(f_\rho^\phi)}.$$

Then Theorem 29 gives the following learning rate.

Corollary 30 *Let ϕ be a loss function such that $\phi''(0)$ exists and is positive. Under the assumptions of Theorem 29, if $\gamma = \frac{2}{5+10p+5\beta}$, we have*

$$\mathbb{E}_{z_1,\dots,z_T}\left(\mathcal{R}(sgn(f_{T+1})) - \mathcal{R}(f_c)\right) \le \widetilde{C}_{\phi,\beta} T^{-\min\{\frac{\beta}{5+10p+5\beta}, \frac{b}{4} - \frac{2+3p}{10+20p+10\beta}\}},$$

where $\tilde{C}_{\phi,\beta}$ is a constant independent of T.

As an example, the q-norm SVM loss $\phi(x) = ((1-x)_+)^q$ with $q > 1$ satisfies $\phi''(0) > 0$ and (17) with $p = q - 1$. So Corollary 30 yields the following rates.

Example 7 *Let $\phi(x) = ((1-x)_+)^q$ with $q > 1$. Under the assumptions of Theorem 29, if $\gamma = \frac{2}{10q-5+5\beta}$, $\alpha = \frac{8q-6+4\beta}{10q-5+5\beta}$ and $b > \frac{6q-2}{10q-5+5\beta}$, then*

$$\mathbb{E}_{z_1,\dots,z_T}\left(\mathcal{R}(sgn(f_{T+1})) - \mathcal{R}(f_c)\right) = O\left(T^{-\min\{\frac{\beta}{10q-5+5\beta}, \frac{b}{4} - \frac{3q-1}{20q-10+10\beta}\}}\right).$$

Finally we verify the learning rates for regression with insensitive loss stated in Section 2.

5.2 Proof of Theorem 10

We need the regularizing function $f_\lambda^{V_{ls}}$ defined by (13) with the least square loss $V = V_{ls}$. It can be found, for example, in Smale and Zhou (2007) that regularity condition (7) implies

$$\|f_\lambda^{V_{ls}} - f_\rho\|_K \le \left(\frac{\lambda}{2}\right)^{r-\frac{1}{2}} \|g_\rho\|_{L^2_{\rho_X}}, \qquad \text{when } \frac{1}{2} < r \le \frac{3}{2}$$

and

$$\|f_\lambda^{V_{ls}} - f_\rho\|_{L^2_{\rho_X}} \le \left(\frac{\lambda}{2}\right)^{r} \|g_\rho\|_{L^2_{\rho_X}}, \qquad \text{when } 0 < r \le 1.$$

It follows that when $\lambda \le 2(\kappa\|g_\rho\|_{L^2_{\rho_X}})^{2/(1-2r)}$, we have $\|f_\lambda^{V_{ls}} - f_\rho\|_{C(X)} \le 1$. Thus by the special form of the conditional distribution ρ_x, we see from the conclusion of Example 6 with $p = 1$ that

$f_\lambda^{V_{in}} = f_{2\lambda}^{V_{ls}}$ and bounds for $\|f_\lambda^{V_{in}} - f_\rho\|_K$ and $\|f_\lambda^{V_{in}} - f_\rho\|_{L^2_{\rho_X}}$ follow. Moreover, condition (12) for $\mathcal{D}(\lambda)$ is valid with $\beta = 1$.

Now we check other conditions of Proposition 28.

Condition (16) is valid because $K \in C^{2s}(X \times X)$ with $0 < s \le \frac{1}{2}$.

By a simple computation, incremental condition (17) is verified with exponent $p = 0$.

Note that $\|f_\rho\|_K \le \kappa^{2r-1}\|g_\rho\|_{L^2_{\rho_X}}$ and $\|f_\rho\|_{C^s(X)} \le (\kappa + \kappa_{2s})\|f_\rho\|_K$. Then for any $x, u \in X$ and $g \in C^s(Y)$, we see from the uniform distribution ρ_x and ρ_u that

$$\begin{aligned}\left|\int_Y g(y)d(\rho_x - \rho_u)\right| &= \frac{1}{2}\left|\int_{f_\rho(x)-1}^{f_\rho(x)+1} g(y)dy - \int_{f_\rho(u)-1}^{f_\rho(u)+1} g(y)dy\right| \le \|g\|_{C(Y)}|f_\rho(x) - f_\rho(u)| \\ &\le \|g\|_{C(Y)}(\kappa + \kappa_{2s})\kappa^{2r-1}\|g_\rho\|_{L^2_{\rho_X}}(d(x,u))^s.\end{aligned}$$

This verifies Lipschitz s continuous condition (4) for $\{\rho_x\}$ with constant $C_\rho = (\kappa + \kappa_{2s})\kappa^{2r-1}\|g_\rho\|_{L^2_{\rho_X}}$.

Thus all conditions of Proposition 28 are satisfied and we obtain

$$\mathbb{E}_{z_1,z_2,\cdots,z_T}(\|f_{T+1} - f_{\lambda_T}^{V_{in}}\|_K^2) \le C_{K,V,\rho,b,\beta,s}T^{-\theta}$$

where

$$\theta := \min\left\{2 - 2\gamma - 2\alpha, \alpha - \gamma, b - 2\gamma\right\}.$$

Finally we get

$$\mathbb{E}_{z_1,z_2,\cdots,z_T}(\|f_{T+1} - f_\rho\|_K) \le \sqrt{C_{K,V,\rho,b,\beta,s}}T^{-\theta/2} + \left(\frac{\lambda_{T+1}}{2}\right)^{r-\frac{1}{2}}\|g_\rho\|_{L^2_{\rho_X}}, \quad \text{when } \frac{1}{2} < r \le \frac{3}{2}$$

and

$$\mathbb{E}_{z_1,z_2,\cdots,z_T}(\|f_{T+1} - f_\rho\|_{L^2_{\rho_X}}) \le \kappa\sqrt{C_{K,V,\rho,b,\beta,s}}T^{-\theta/2} + \left(\frac{\lambda_{T+1}}{2}\right)^{r}\|g_\rho\|_{L^2_{\rho_X}}, \quad \text{when } 0 < r \le 1.$$

Then our desired learning rates follow. ■

Acknowledgments

We would like to thank the referees for their constructive suggestions and comments. The work described in this paper was partially supported by a grant from the Research Grants Council of Hong Kong [Project No. CityU 104007] and National Science Fund for Distinguished Young Scholars of China [Project No. 10529101]. The corresponding author is Ding-Xuan Zhou.

Appendix A.

This appendix includes some detailed proofs.

A.1 Proof of Proposition 6

The first statement follows by taking $g(y) = y$ on Y because

$$|f_\rho(x) - f_\rho(u)| = \left| \int_Y g(y) d(\rho_x - \rho_u)(y) \right| \le \|\rho_x - \rho_u\|_{(C^s(Y))^*} \|g\|_{C^s(Y)}$$

and $\|g\|_{C^s(Y)} = \|g\|_{C(Y)} + |g|_{C^s(Y)} \le \sup_{y\in Y} |y| + 2^{1-s} \sup_{y\in Y} |y|$. Actually the above estimates tell us that f_ρ is continuous and belongs to $C^s(X)$ with $|f_\rho|_{C^s(X)} \le C_\rho(1 + 2^{1-s}) \sup_{y\in Y} |y|$.

For the second statement, since $Y = \{1, -1\}$, we have $f_\rho(x) = \rho_x(\{1\}) - \rho_x(\{-1\})$. It follows that for each $y \in Y$ and $x \in X$, there holds $\rho_x(\{y\}) = \frac{1+yf_\rho(x)}{2}$. So for any $g \in C^s(Y)$ and $x, u \in X$,

$$\int_Y g(y) d(\rho_x - \rho_u)(y) = \sum_{y\in Y} g(y) \frac{y[f_\rho(x) - f_\rho(u)]}{2} = \sum_{y\in Y} \frac{yg(y)}{2} [f_\rho(x) - f_\rho(u)].$$

Now the conclusion follows from

$$\left| \int_Y g(y) d(\rho_x - \rho_u)(y) \right| \le \|g\|_{C(Y)} |f_\rho(x) - f_\rho(u)| \le |f_\rho|_{C^s(X)} (d(x,u))^s \|g\|_{C^s(Y)}.$$

This proves Proposition 6. ■

A.2 Proof of Example 4

For $x \in X$, we have $\rho_x(\{1\}) = f_{\rho,-1}(x) + f_\rho(x)$ and $\rho_x(\{0\}) = 1 - 2f_{\rho,-1}(x) - f_\rho(x)$. Hence for any $g \in C^s(Y)$,

$$\int_Y g(y) d\rho_x = f_{\rho,-1}(x) \{g(1) - 2g(0) + g(-1)\} + f_\rho(x) \{g(1) - g(0)\} + g(0)$$

and for $u \in X$,

$$\int_Y g(y) d(\rho_x - \rho_u) = [f_{\rho,-1}(x) - f_{\rho,-1}(u)] \{g(1) - 2g(0) + g(-1)\} + [f_\rho(x) - f_\rho(u)] \{g(1) - g(0)\}.$$

Then our statement follows from the first part of Proposition 6. This proves the conclusion of Example 4. ■

A.3 Proof of Example 6

Let $x \in X$. When $f(x) \ge f_\rho^{V_{in}}(x)$, we see from the explicit form of the insensitive loss V_{in} that

$$\begin{aligned} & \int_Y V_{in}(y, f(x)) d\rho_x(y) - \int_Y V_{in}(y, f_\rho^{V_{in}}(x)) d\rho_x(y) \\ = & \int_{y \ge f(x)} f_\rho^{V_{in}}(x) - f(x) d\rho_x + \int_{y \le f_\rho^{V_{in}}(x)} f(x) - f_\rho^{V_{in}}(x) d\rho_x \\ & + \int_{f_\rho^{V_{in}}(x) < y < f(x)} f(x) + f_\rho^{V_{in}}(x) - 2y d\rho_x. \end{aligned}$$

It follows that

$$\mathcal{E}(f) - \mathcal{E}(f_\rho^{V_{in}}) = \int_X \left\{ |f(x) - f_\rho^{V_{in}}(x)| \Delta_x + 2 \int_{I_x} |f(x) - y| d\rho_x \right\} d\rho_X \tag{31}$$

where

$$\Delta_x := \left|\rho_x\left(\{y \in Y : y \le f_\rho^{V_{in}}(x)\}\right) - \rho_x\left(\{y \in Y : y > f_\rho^{V_{in}}(x)\}\right)\right|$$

and I_x in the open interval between $f_\rho^{V_{in}}(x)$ and $f(x)$. The same relation (31) also holds when $f(x) < f_\rho^{V_{in}}(x)$.

Now we use the special assumption on the conditional distributions and see that the median and mean of ρ_x are equal: $f_\rho^{V_{in}}(x) = f_\rho(x)$ for each $x \in X$. Moreover, $\Delta_x = 0$ and when $f_\rho(x) \le f(x) \le f_\rho(x) + 1$, we have

$$2\int_{I_x} |f(x) - y| d\rho_x = 2\int_0^{f(x)-f_\rho(x)} (f(x) - f_\rho(x) - u)\frac{p}{2}u^{p-1}du = \frac{|f(x) - f_\rho(x)|^{p+1}}{p+1}.$$

The same expression holds true when $f_\rho(x) - 1 \le f(x) < f_\rho(x)$. When $|f(x) - f_\rho(x)| > 1$, since ρ_x vanishes outside $[-f_\rho(x) - 1, f_\rho(x) + 1]$, we have $2\int_{I_x} |f(x) - y| d\rho_x = |f(x) - f_\rho(x)| - \frac{p}{p+1}$. Therefore, (31) is the same as

$$\begin{aligned}\mathcal{E}(f) - \mathcal{E}(f_\rho^{V_{in}}) &= \int_{\{x \in X : |f(x) - f_\rho(x)| \le 1\}} \frac{|f(x) - f_\rho(x)|^{p+1}}{p+1} d\rho_X \\ &\quad + \int_{\{x \in X : |f(x) - f_\rho(x)| > 1\}} |f(x) - f_\rho(x)| - \frac{p}{p+1} d\rho_X.\end{aligned}$$

This proves the desired expression for $\mathcal{E}(f) - \mathcal{E}(f_\rho^{V_{in}})$ and hence the bound for $\mathcal{D}(\lambda)$. ■

References

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101:138–156, 2006.

A. Caponnetto, L. Rosasco and Y. Yao. On early stopping in gradient descent learning. *Constructive Approximation*, 26:289–315, 2007.

D. R. Chen, Q. Wu, Y. Ying and D. X. Zhou. Support vector machine soft margin classifiers: error analysis. *Journal of Machine Learning Research*, 5:1143–1175, 2004.

N. Cesa-Bianchi, P. Long and M. K. Warmuth. Worst-case quadratic loss bounds for prediction using linear functions and gradient descent. *IEEE Transactions on Neural Networks*, 7:604–619, 1996.

N. Cesa-Bianchi, A. Conconi and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50:2050-2057, 2004.

E. De Vito, A. Caponnetto, and L. Rosasco. Model selection for regularized least-squares algorithm in learning theory. *Foundations of Computational Mathematics*, 5:59–85, 2005.

E. De Vito, L. Rosasco, A. Caponnetto, M. Piana, and A. Verri. Some properties of regularized kernel methods. *Journal of Machine Learning Research*, 5:1363–1390, 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. *Advances in Computational Mathematics*, 13:1–50, 2000.

D. Hardin, I. Tsamardinos, and C. F. Aliferis. A theoretical characterization of linear SVM-based feature selection. *Proc. of the 21st Int. Conf. on Machine Learning*, Banff, Canada, 2004.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Trans. Signal Processing* 52:2165–2176, 2004.

S. Mukherjee and Q. Wu. Estimation of gradients and coordinate covariation in classification. *Journal of Machine Learning Research*, 7:2481–2514, 2006.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, 1970.

S. Smale and Y. Yao. Online learning algorithms. *Foundations of Computational Mathematics*, 6:145–170, 2006.

S. Smale and D.X. Zhou. Estimating the approximation error in learning theory. *Analysis and Applications*, 1:17–41, 2003.

S. Smale and D.X. Zhou. Shannon sampling and function reconstruction from point values. *Bulletin of the American Mathematical Society*, 41:279–305, 2004.

S. Smale and D.X. Zhou. Learning theory estimates via integral operators and their applications. *Constructive Approximation*, 26:153–172, 2007.

S. Smale and D. X. Zhou. Online learning with Markov sampling. *Analysis and Applications*, 7:87–113, 2009.

I. Steinwart. Support vector machines are universally consistent. *Journal of Complexity*, 18:768–791, 2002.

I. Steinwart, D. Hush and C. Scovel. Learning from dependent observations. *Journal of Multivariate Analysis*, 100:175–194, 2009.

P. Tarrès and Y. Yao. Online learning as stochastic approximations of regularization paths. preprint, 2005.

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

Q. Wu, Y. Ying and D. X. Zhou. Multi-kernel regularized classifiers. *Journal of Complexity*, 23:108–134, 2007.

Q. Wu, Y. Ying and D. X. Zhou. Learning theory: From regression to classification. *Topics in Multivariate Approximation and Interpolation* (K. Jetter et al. eds.), Elsivier, pp. 257–290, 2006.

G. B. Ye and D. X. Zhou. Fully online classification by regularization. *Applied and Computational Harmonic Analysis*, 23:198–214, 2007.

Y. Ying. Convergence analysis of online algorithms. *Advances in Computational Mathematics*, 27:273–291, 2007.

Y. Ying and D.X. Zhou. Online regularized classification algorithms. *IEEE Transactions on Information Theory*, 52:4775–4788, 2006.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32:56–85, 2004.

D. X. Zhou. The covering number in learning theory. *Journal of Complexity*, 18:739–767, 2002.

D. X. Zhou. Capacity of reproducing kernel spaces in learning theory. *IEEE Transactions on Information Theory*, 49:1743–1752, 2003.

D. X. Zhou. Derivative reproducing properties for kernel methods in learning theory. *Journal of Computational and Applied Mathematics*, 220:456–463, 2008.

Journal of Machine Learning Research 10 (2009) 2899-2934 Submitted 2/09; Revised 7/09; Published 12/09

Efficient Online and Batch Learning Using Forward Backward Splitting

John Duchi JDUCHI@CS.BERKELEY.EDU
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720 USA

Yoram Singer SINGER@GOOGLE.COM
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043 USA

Editor: Yoav Freund

Abstract

We describe, analyze, and experiment with a framework for empirical loss minimization with regularization. Our algorithmic framework alternates between two phases. On each iteration we first perform an *unconstrained* gradient descent step. We then cast and solve an instantaneous optimization problem that trades off minimization of a regularization term while keeping close proximity to the result of the first phase. This view yields a simple yet effective algorithm that can be used for batch penalized risk minimization and online learning. Furthermore, the two phase approach enables sparse solutions when used in conjunction with regularization functions that promote sparsity, such as ℓ_1. We derive concrete and very simple algorithms for minimization of loss functions with ℓ_1, ℓ_2, ℓ_2^2, and ℓ_∞ regularization. We also show how to construct efficient algorithms for mixed-norm ℓ_1/ℓ_q regularization. We further extend the algorithms and give efficient implementations for very high-dimensional data with sparsity. We demonstrate the potential of the proposed framework in a series of experiments with synthetic and natural data sets.

Keywords: subgradient methods, group sparsity, online learning, convex optimization

1. Introduction

Before we begin, we establish notation for the content of this paper. We denote scalars by lower case letters and vectors by lower case bold letters, for example, $\boldsymbol{w}$. The inner product of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is denoted $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$. We use $\|\boldsymbol{x}\|_p$ to denote the p-norm of the vector $\boldsymbol{x}$ and $\|\boldsymbol{x}\|$ as a shorthand for $\|\boldsymbol{x}\|_2$.

The focus of this paper is an algorithmic framework for regularized convex programming to minimize the following sum of two functions:

$$f(\boldsymbol{w}) + r(\boldsymbol{w}), \tag{1}$$

where both f and r are convex bounded below functions (so without loss of generality we assume they are into $\mathbb{R}_+$). Often, the function f is an empirical loss and takes the form $\sum_{i \in S} \ell_i(\boldsymbol{w})$ for a sequence of loss functions $\ell_i : \mathbb{R}^n \to \mathbb{R}_+$, and $r(\boldsymbol{w})$ is a regularization term that penalizes for excessively complex vectors, for instance $r(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|_p$. This task is prevalent in machine learning, in which a learning problem for decision and prediction problems is cast as a convex optimization problem. To that end, we investigate a general and intuitive algorithm, known as *forward-backward splitting*, to minimize Eq. (1), focusing especially on derivations for and use of non-differentiable regularization functions.

Many methods have been proposed to minimize general convex functions such as that in Eq. (1). One of the most general is the subgradient method (see, e.g., Bertsekas, 1999), which is elegant and very simple. Let $\partial f(\boldsymbol{w})$ denote the subgradient set of f at $\boldsymbol{w}$, namely,

$$\partial f(\boldsymbol{w}) = \left\{ \boldsymbol{g} \mid \forall \boldsymbol{v} : f(\boldsymbol{v}) \geq f(\boldsymbol{w}) + \langle \boldsymbol{g}, \boldsymbol{v} - \boldsymbol{w} \rangle \right\}.$$

Sub-gradient procedures then minimize the function $f(\boldsymbol{w})$ by iteratively updating the parameter vector $\boldsymbol{w}$ according to the update rule

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f \quad ,$$

where η_t is a constant or diminishing a step size and $\boldsymbol{g}_t^f \in \partial f(\boldsymbol{w}_t)$ is an arbitrary vector from the subgradient set of f evaluated at $\boldsymbol{w}_t$. A more general method than the above is the projected gradient method, which iterates

$$\boldsymbol{w}_{t+1} \; = \; \Pi_\Omega\left(\boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f\right) \; = \; \operatorname*{argmin}_{\boldsymbol{w}\in\Omega} \left\{ \left\| \boldsymbol{w} - (\boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f) \right\|_2^2 \right\}$$

where $\Pi_\Omega(\boldsymbol{w})$ is the Euclidean projection of $\boldsymbol{w}$ onto the set Ω. Standard results (Bertsekas, 1999) show that the (projected) subgradient method converges at a rate of $O(1/\varepsilon^2)$, or equivalently that the error $f(\boldsymbol{w}) - f(\boldsymbol{w}^\star) = O(1/\sqrt{T})$, given some simple assumptions on the boundedness of the subdifferential set and Ω (we have omitted constants dependent on $\|\partial f\|$ or $\dim(\Omega)$).

If we use the subgradient method to minimize Eq. (1), the iterates are simply $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f - \eta_t \boldsymbol{g}_t^r$, where $\boldsymbol{g}_t^r \in \partial r(\boldsymbol{w}_t)$. A common problem in subgradient methods is that if r or f is non-differentiable, the iterates of the subgradient method are very rarely at the points of non-differentiability. In the case of regularization functions such as $r(\boldsymbol{w}) = \|\boldsymbol{w}\|_1$, however, these points (zeros in the case of the ℓ_1-norm) are often the true minima of the function. Furthermore, with ℓ_1 and similar penalties, zeros are desirable solutions as they tend to convey information about the structure of the problem being solved, and in the case of statistical inference, can often yield the correct sparsity structure of the parameters (Zhao and Yu, 2006; Meinshausen and Bühlmann, 2006).

There has been a significant amount of work related to minimizing Eq. (1), especially when the function r is a sparsity-promoting regularizer, and much of it stems from the machine learning, statistics, and optimization communities. We can hardly do justice to the body of prior work, and we provide a few references here to the research we believe is most directly related. The approach we pursue below is known as *forward-backward splitting* in the optimization literature, which is closely related to the proximal method.

The forward-backward splitting method was first proposed by Lions and Mercier (1979) and has been analyzed by several researches in the context of maximal monotone operators in the optimization literature. Chen and Rockafellar (1997) and Tseng (2000) give conditions and modifications of forward-backward splitting to attain linear convergence rates. Combettes and Wajs (2005) give proofs of convergence for forward-backward splitting in Hilbert spaces under asymptotically negligible perturbations, though without establishing strong rates of convergence. Prior work on convergence of the method often requires an assumption of strong monotonicity of the maximal monotone operators (equivalent to strong convexity of at least one of the functions in Eq. (1)), and as far as we know, all analyses assume that f is differentiable with Lipschitz-continuous gradient. The analyses have also been carried out in a non-stochastic and non-online setting.

More recently, Wright et al. (2009) suggested the use of the method for sparse signal reconstruction, where $f(\boldsymbol{w}) = \|\boldsymbol{y} - A\boldsymbol{w}\|^2$, though they note that the method can apply to suitably smooth convex functions f. Nesterov (2007) gives analysis of convergence rates using gradient mapping techniques when f has Lipschitz continuous gradient, which was inspired by Wright et al. In the special case that $r(\boldsymbol{w}) = \lambda\|\boldsymbol{w}\|_1$, similar methods to the algorithms we investigate have been proposed and termed iterative thresholding (Daubechies et al., 2004) or truncated gradient (Langford et al., 2008) in signal processing and machine learning, but the authors were apparently unaware of the connection to splitting methods.

Similar projected-gradient methods, when the regularization function r is no longer part of the objective function but rather cast as a constraint so that $r(\boldsymbol{w}) \leq \lambda$, are also well known (Bertsekas, 1999). In signal processing, the problem is often termed as an inverse problem with sparsity constraints, see for example, Daubechies et al. (2008) and the references therein. Duchi et al. (2008) give a general and efficient projected gradient method for ℓ_1-constrained problems. We make use of one of Duchi et al.'s results in obtaining an efficient algorithm for the case when $r(\boldsymbol{w}) = \|\boldsymbol{w}\|_\infty$ (a setting useful for mixed-norm regularization). There is also a body of literature on regret analysis for online learning and online convex programming with convex constraints, which we build upon here (Zinkevich, 2003; Hazan et al., 2006; Shalev-Shwartz and Singer, 2007). Learning sparse models generally is of great interest in the statistics literature, specifically in

the context of consistency and recovery of sparsity patterns through ℓ_1 or mixed-norm regularization across multiple tasks (Meinshausen and Bühlmann, 2006; Obozinski et al., 2008; Zhao et al., 2006).

In this paper, we describe a general gradient-based framework for online and batch convex programming. To make our presentation a little simpler, we call our approach FOBOS, for FOrward-Backward Splitting.[1] Our proofs are made possible through the use of "forward-looking" subgradients, and FOBOS is a distillation of some the approaches mentioned above for convex programming. Our alternative view lends itself to unified analysis and more general settings, efficient implementation, and provides a flexible tool for the derivation of algorithms for old and new convex programming settings.

The paper is organized as follows. In the next section, we begin by introducing and formally defining the method, giving some simple preliminary analysis. We follow the introduction by giving in Sec. 3 rates of convergence for batch (offline) optimization. We then extend the results to stochastic gradient descent and provide regret bounds for online convex programming in Sec. 4. To demonstrate the simplicity and usefulness of the framework, we derive in Sec. 5 algorithms for several different choices of the regularizing function r, though most of these results are known. We then extend these methods to be efficient in very high dimensional learning settings where the input data is sparse in Sec. 6. Finally, we conclude in Sec. 7 with experiments examining various aspects of the proposed framework, in particular the runtime and sparsity selection performance of the derived algorithms.

2. Forward-Looking Subgradients and Forward-Backward Splitting

Our approach to Forward-Backward Splitting is motivated by the desire to have the iterates $\boldsymbol{w}_t$ attain points of non-differentiability of the function r. The method alleviates the problems of non-differentiability in cases such as ℓ_1-regularization by taking analytical minimization steps interleaved with subgradient steps. Put informally, FOBOS can be viewed as analogous to the *projected* subgradient method while replacing or augmenting the projection step with an instantaneous minimization problem for which it is possible to derive a closed form solution. FOBOS is succinct as each iteration consists of the following two steps:

$$\boldsymbol{w}_{t+\frac{1}{2}} = \boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f, \tag{2}$$

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \frac{1}{2} \left\| \boldsymbol{w} - \boldsymbol{w}_{t+\frac{1}{2}} \right\|^2 + \eta_{t+\frac{1}{2}} r(\boldsymbol{w}) \right\} . \tag{3}$$

In the above, $\boldsymbol{g}_t^f$ is a vector in $\partial f(\boldsymbol{w}_t)$ and η_t is the step size at time step t of the algorithm. The actual value of η_t depends on the specific setting and analysis. The first step thus simply amounts to an unconstrained subgradient step with respect to the function f. In the second step we find a new vector that interpolates between two goals: (i) stay close to the interim vector $\boldsymbol{w}_{t+\frac{1}{2}}$, and (ii) attain a low complexity value as expressed by r. Note that the regularization function is scaled by an interim step size, denoted $\eta_{t+\frac{1}{2}}$. The analyses we describe in the sequel determine the specific value of $\eta_{t+\frac{1}{2}}$, which is either η_t or η_{t+1}.

A key property of the solution of Eq. (3) is the necessary condition for optimality and gives the reason behind the name FOBOS. Namely, the zero vector must belong to subgradient set of the objective at the optimum $\boldsymbol{w}_{t+1}$, that is,

$$\mathbf{0} \in \partial \left\{ \frac{1}{2} \left\| \boldsymbol{w} - \boldsymbol{w}_{t+\frac{1}{2}} \right\|^2 + \eta_{t+\frac{1}{2}} r(\boldsymbol{w}) \right\} \Bigg|_{\boldsymbol{w}=\boldsymbol{w}_{t+1}} .$$

Since $\boldsymbol{w}_{t+\frac{1}{2}} = \boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f$, the above property amounts to

$$\mathbf{0} \in \boldsymbol{w}_{t+1} - \boldsymbol{w}_t + \eta_t \boldsymbol{g}_t^f + \eta_{t+\frac{1}{2}} \partial r(\boldsymbol{w}_{t+1}). \tag{4}$$

1. An earlier draft of this paper referred to our algorithm as FOLOS, for FOrward LOoking Subgradients. In order not to confuse readers of the early draft, we attempt to stay close to the earlier name and use the acronym FOBOS rather than Fobas.

The property $\mathbf{0} \in \boldsymbol{w}_{t+1} - \boldsymbol{w}_t + \eta_t \boldsymbol{g}_t^f + \eta_{t+\frac{1}{2}} \partial r(\boldsymbol{w}_{t+1})$ implies that so long as we choose $\boldsymbol{w}_{t+1}$ to be the minimizer of Eq. (3), we are guaranteed to obtain a vector $\boldsymbol{g}_{t+1}^r \in \partial r(\boldsymbol{w}_{t+1})$ such that

$$\mathbf{0} = \boldsymbol{w}_{t+1} - \boldsymbol{w}_t + \eta_t \boldsymbol{g}_t^f + \eta_{t+\frac{1}{2}} \boldsymbol{g}_{t+1}^r \ .$$

The above equation can be understood as an update scheme where the new weight vector $\boldsymbol{w}_{t+1}$ is a linear combination of the previous weight vector $\boldsymbol{w}_t$, a vector from the subgradient set of f evaluated at $\boldsymbol{w}_t$, and a vector from the subgradient of r evaluated at the yet to be determined $\boldsymbol{w}_{t+1}$, hence the name Forward-Looking Subgradient. To recap, we can write $\boldsymbol{w}_{t+1}$ as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \, \boldsymbol{g}_t^f - \eta_{t+\frac{1}{2}} \, \boldsymbol{g}_{t+1}^r, \tag{5}$$

where $\boldsymbol{g}_t^f \in \partial f(\boldsymbol{w}_t)$ and $\boldsymbol{g}_{t+1}^r \in \partial r(\boldsymbol{w}_{t+1})$. Solving Eq. (3) with r above has two main benefits. First, from an algorithmic standpoint, it enables sparse solutions at virtually no additional computational cost. Second, the forward-looking gradient allows us to build on existing analyses and show that the resulting framework enjoys the formal convergence properties of many existing gradient-based and online convex programming algorithms.

3. Convergence Analysis of FOBOS

Upon first look FOBOS looks substantially different from sub-gradient and online convex programming methods. However, the fact that FOBOS actually employs a forward-looking subgradient of the regularization function lets us build nicely on existing analyses. In this section we modify known results while using the forward-looking property of FOBOS to provide convergence rate analysis for FOBOS. To do so we will set $\eta_{t+\frac{1}{2}}$ properly. As we show in the sequel, it is sufficient to set $\eta_{t+\frac{1}{2}}$ to η_t or η_{t+1}, depending on whether we are doing online or batch optimization, in order to obtain convergence and low regret bounds. We start with an analysis of FOBOS in a batch setting. In this setting we use the subgradient of f, set $\eta_{t+\frac{1}{2}} = \eta_{t+1}$ and update $\boldsymbol{w}_t$ to $\boldsymbol{w}_{t+1}$ as prescribed by Eq. (2) and Eq. (3).

Throughout the section we denote by $\boldsymbol{w}^\star$ the minimizer of $f(\boldsymbol{w}) + r(\boldsymbol{w})$. In what follows, define $\|\partial f(\boldsymbol{w})\| \triangleq \sup_{\boldsymbol{g} \in \partial f(\boldsymbol{w})} \|\boldsymbol{g}\|$. We begin by deriving convergence results under the fairly general assumption (see, e.g., Langford et al. 2008 or Shalev-Shwartz and Tewari 2009) that the subgradients are bounded as follows:

$$\|\partial f(\boldsymbol{w})\|^2 \leq A f(\boldsymbol{w}) + G^2, \quad \|\partial r(\boldsymbol{w})\|^2 \leq A r(\boldsymbol{w}) + G^2 \ . \tag{6}$$

For example, any Lipschitz loss (such as the logistic loss or hinge loss used in support vector machines) satisfies the above with $A = 0$ and G equal to the Lipschitz constant. Least squares estimation satisfies Eq. (6) with $G = 0$ and $A = 4$. The next lemma, while technical, provides a key tool for deriving all of the convergence results in this paper.

Lemma 1 (Bounding Step Differences) *Assume that the norms of the subgradients of the functions f and r are bounded as in Eq. (6):*

$$\|\partial f(\boldsymbol{w})\|^2 \leq A f(\boldsymbol{w}) + G^2, \quad \|\partial r(\boldsymbol{w})\|^2 \leq A r(\boldsymbol{w}) + G^2 \ .$$

Let $\eta_{t+1} \leq \eta_{t+\frac{1}{2}} \leq \eta_t$ and suppose that $\eta_t \leq 2\eta_{t+\frac{1}{2}}$. If we use the FOBOS *update of Eqs. (2) and (3), then for a constant $c \leq 4$ and any vector $\boldsymbol{w}^\star$,*

$$\begin{aligned} & 2\eta_t(1 - cA\eta_t) f(\boldsymbol{w}_t) + 2\eta_{t+\frac{1}{2}}(1 - cA\eta_{t+\frac{1}{2}}) r(\boldsymbol{w}_{t+1}) \\ & \quad \leq \quad 2\eta_t f(\boldsymbol{w}^\star) + 2\eta_{t+\frac{1}{2}} r(\boldsymbol{w}^\star) + \|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{w}^\star\|^2 + 7\eta_t \eta_{t+\frac{1}{2}} G^2 \ . \end{aligned}$$

Proof We begin with a few simple properties of the forward-looking subgradient steps before proceeding with the core of the proof. Note first that for some $\boldsymbol{g}_t^f \in \partial f(\boldsymbol{w}_t)$ and $\boldsymbol{g}_{t+1}^r \in \partial r(\boldsymbol{w}_{t+1})$, we have as in Eq. (5)

$$\boldsymbol{w}_{t+1} - \boldsymbol{w}_t = -\eta_t \boldsymbol{g}_t^f - \eta_{t+\frac{1}{2}} \boldsymbol{g}_{t+1}^r \ . \tag{7}$$

The definition of a subgradient implies that for any $\boldsymbol{g}_{t+1}^r \in \partial r(\boldsymbol{w}_{t+1})$ (and similarly for any $\boldsymbol{g}_t^f \in \partial f(\boldsymbol{w}_t)$ with $f(\boldsymbol{w}_t)$ and $f(\boldsymbol{w}^\star)$)

$$r(\boldsymbol{w}^\star) \geq r(\boldsymbol{w}_{t+1}) + \left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}^\star - \boldsymbol{w}_{t+1} \right\rangle \quad \Rightarrow \quad -\left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_{t+1} - \boldsymbol{w}^\star \right\rangle \leq r(\boldsymbol{w}^\star) - r(\boldsymbol{w}_{t+1}). \tag{8}$$

From the Cauchy-Shwartz Inequality and Eq. (7), we obtain

$$\begin{aligned}
\left\langle \boldsymbol{g}_{t+1}^r, (\boldsymbol{w}_{t+1} - \boldsymbol{w}_t) \right\rangle &= \left\langle \boldsymbol{g}_{t+1}^r, (-\eta_t \boldsymbol{g}_t^f - \eta_{t+\frac{1}{2}} \boldsymbol{g}_{t+1}^r) \right\rangle \\
&\leq \|\boldsymbol{g}_{t+1}^r\| \|\eta_{t+\frac{1}{2}} \boldsymbol{g}_{t+1}^r + \eta_t \boldsymbol{g}_t^f\| \leq \eta_{t+\frac{1}{2}} \|\boldsymbol{g}_{t+1}^r\|^2 + \eta_t \|\boldsymbol{g}_{t+1}^r\| \|\boldsymbol{g}_t^f\| \\
&\leq \eta_{t+\frac{1}{2}} \left(A r(\boldsymbol{w}_{t+1}) + G^2 \right) + \eta_t \left(A \max\{f(\boldsymbol{w}_t), r(\boldsymbol{w}_{t+1})\} + G^2 \right) \ .
\end{aligned} \tag{9}$$

We now proceed to bound the difference between $\boldsymbol{w}^\star$ and $\boldsymbol{w}_{t+1}$, and using a telescoping sum we will eventually bound $f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t) - f(\boldsymbol{w}^\star) - r(\boldsymbol{w}^\star)$. First, we expand norm squared of the difference between $\boldsymbol{w}_t$ and $\boldsymbol{w}_{t+1}$,

$$\begin{aligned}
\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^\star\|^2 &= \|\boldsymbol{w}_t - (\eta_t \boldsymbol{g}_t^f + \eta_{t+\frac{1}{2}} \boldsymbol{g}_{t+1}^r) - \boldsymbol{w}^\star\|^2 \\
&= \|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2 - 2\left[\eta_t \left\langle \boldsymbol{g}_t^f, \boldsymbol{w}_t - \boldsymbol{w}^\star \right\rangle + \eta_{t+\frac{1}{2}} \left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_t - \boldsymbol{w}^\star \right\rangle \right] + \|\eta_t \boldsymbol{g}_t^f + \eta_{t+\frac{1}{2}} \boldsymbol{g}_{t+1}^r\|^2 \\
&= \|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2 - 2\eta_t \left\langle \boldsymbol{g}_t^f, \boldsymbol{w}_t - \boldsymbol{w}^\star \right\rangle + \|\eta_t \boldsymbol{g}_t^f + \eta_{t+\frac{1}{2}} \boldsymbol{g}_{t+1}^r\|^2 \\
&\quad -2\eta_{t+\frac{1}{2}} \left[\left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_{t+1} - \boldsymbol{w}^\star \right\rangle - \left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_{t+1} - \boldsymbol{w}_t \right\rangle \right] \ .
\end{aligned} \tag{10}$$

We can bound the third term above by noting that

$$\begin{aligned}
&\|\eta_t \boldsymbol{g}_t^f + \eta_{t+\frac{1}{2}} \boldsymbol{g}_{t+1}^r\|^2 \\
&= \eta_t^2 \|\boldsymbol{g}_t^f\|^2 + 2\eta_t \eta_{t+\frac{1}{2}} \left\langle \boldsymbol{g}_t^f, \boldsymbol{g}_{t+1}^r \right\rangle + \eta_{t+\frac{1}{2}}^2 \|\boldsymbol{g}_{t+1}^r\|^2 \\
&\leq \eta_t^2 A f(\boldsymbol{w}_t) + 2\eta_t \eta_{t+\frac{1}{2}} A \max\{f(\boldsymbol{w}_t), r(\boldsymbol{w}_{t+1})\} + \eta_{t+\frac{1}{2}}^2 A r(\boldsymbol{w}_{t+1}) + 4\eta_t^2 G^2 \ .
\end{aligned}$$

We now use Eq. (9) to bound the last term of Eq. (10) and the above bound on $\eta_t \boldsymbol{g}_t^f + \eta_{t+\frac{1}{2}} \boldsymbol{g}_{t+1}^r$ to get that

$$\begin{aligned}
&\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^\star\|^2 \\
&\leq \|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2 - 2\eta_t \left\langle \boldsymbol{g}_t^f, \boldsymbol{w}_t - \boldsymbol{w}^\star \right\rangle - 2\eta_{t+\frac{1}{2}} \left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_{t+1} - \boldsymbol{w}^\star \right\rangle + \|\eta_t \boldsymbol{g}_t^f + \eta_{t+\frac{1}{2}} \boldsymbol{g}_{t+1}^r\|^2 \\
&\quad + 2\eta_{t+\frac{1}{2}} \left(\eta_{t+\frac{1}{2}} A r(\boldsymbol{w}_{t+1}) + 2\eta_t A \max\{f(\boldsymbol{w}_t), r(\boldsymbol{w}_{t+1})\} + 2\eta_t G^2 \right) \\
&\leq \|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2 + 2\eta_t \left(f(\boldsymbol{w}^\star) - f(\boldsymbol{w}_t) \right) + 2\eta_{t+\frac{1}{2}} \left(r(\boldsymbol{w}^\star) - r(\boldsymbol{w}_t) \right) + 7\eta_t^2 G^2 \\
&\quad + \eta_t^2 A f(\boldsymbol{w}_t) + 3\eta_t \eta_{t+\frac{1}{2}} A \max\{f(\boldsymbol{w}_t), r(\boldsymbol{w}_t)\} + 2\eta_{t+\frac{1}{2}}^2 A r(\boldsymbol{w}_{t+1}) \qquad (11) \\
&\leq \|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2 + 7\eta_t^2 G^2 \\
&\quad + 2\eta_t \left(f(\boldsymbol{w}^\star) - (1 - c\eta_t A) f(\boldsymbol{w}_t) \right) + 2\eta_{t+\frac{1}{2}} \left(r(\boldsymbol{w}^\star) - (1 - c\eta_{t+\frac{1}{2}} A) r(\boldsymbol{w}_{t+1}) \right) \ . \qquad (12)
\end{aligned}$$

To obtain Eq. (11) we used the standard convexity bounds established earlier in Eq. (8). The final bound given by Eq. (12) is due to the fact that $3A\eta_t \eta_{t+\frac{1}{2}} \leq 6A\eta_t^2$ and that for any $a, b \geq 0$, $\max\{a, b\} \leq a + b$. Rearranging

the terms $f(\cdot)$ and $r(\cdot)$ yields the desired inequality. ■

The lemma allows a proof of the following theorem, which constitutes the basis for deriving concrete convergence results for FOBOS. It is also demonstrates the ease of proving convergence results based on the lemma and the forward looking property.

Theorem 2 *Assume the following hold: (i) the norm of any subgradient from ∂f and the norm of any subgradient from ∂r are bounded as in Eq. (6), (ii) the norm of $\boldsymbol{w}^\star$ is less than or equal to D, (iii) $r(\mathbf{0}) = 0$, and (iv) $\frac{1}{2}\eta_t \leq \eta_{t+1} \leq \eta_t$. Then for a constant $c \leq 4$ with $\boldsymbol{w}_1 = \mathbf{0}$ and $\eta_{t+\frac{1}{2}} = \eta_{t+1}$,*

$$\sum_{t=1}^{T} \left[\eta_t \left((1 - cA\eta_t) f(\boldsymbol{w}_t) - f(\boldsymbol{w}^\star)\right) + \eta_t \left((1 - cA\eta_t) r(\boldsymbol{w}_t) - r(\boldsymbol{w}^\star)\right)\right] \leq D^2 + 7G^2 \sum_{t=1}^{T} \eta_t^2 \ .$$

Proof Rearranging the $f(\boldsymbol{w}^\star)$ and $r(\boldsymbol{w}^\star)$ terms from the bound in Lemma 1, we sum the loss terms over t from 1 through T and get a canceling telescoping sum:

$$\begin{aligned}
&\sum_{t=1}^{T} \left[\eta_t \left((1 - cA\eta_t) f(\boldsymbol{w}_t) - f(\boldsymbol{w}^\star)\right) + \eta_{t+1} \left((1 - cA\eta_{t+1}) r(\boldsymbol{w}_{t+1}) - r(\boldsymbol{w}^\star)\right)\right] \\
&\quad \leq \quad \|\boldsymbol{w}_1 - \boldsymbol{w}^\star\|^2 - \|\boldsymbol{w}_{T+1} - \boldsymbol{w}^\star\|^2 + 7G^2 \sum_{t=1}^{T} \eta_t^2 \ \leq \ \|\boldsymbol{w}_1 - \boldsymbol{w}^\star\|^2 + 7G^2 \sum_{t=1}^{T} \eta_t^2 \ . \qquad (13)
\end{aligned}$$

We now bound the time-shifted $r(\boldsymbol{w}_{t+1})$ terms by noting that

$$\begin{aligned}
&\sum_{t=1}^{T} \eta_{t+1} \left((1 - cA\eta_{t+1}) r(\boldsymbol{w}_{t+1}) - r(\boldsymbol{w}^\star)\right) \\
&\quad = \quad \sum_{t=1}^{T} \eta_t \left((1 - cA\eta_t) r(\boldsymbol{w}_t) - r(\boldsymbol{w}^\star)\right) + \eta_{T+1} \left((1 - cA\eta_{T+1}) r(\boldsymbol{w}_{t+1}) - r(\boldsymbol{w}^\star)\right) + \eta_1 r(\boldsymbol{w}^\star) \\
&\quad \geq \quad \sum_{t=1}^{T} \eta_t \left((1 - cA\eta_t) r(\boldsymbol{w}_{t+1}) - r(\boldsymbol{w}^\star)\right) + r(\boldsymbol{w}^\star)(\eta_1 - \eta_{T+1}) \\
&\quad \geq \quad \sum_{t=1}^{T} \eta_t \left((1 - cA\eta_t) r(\boldsymbol{w}_t) - r(\boldsymbol{w}^\star)\right) \ . \qquad (14)
\end{aligned}$$

Finally, we use the fact that $\|\boldsymbol{w}_1 - \boldsymbol{w}^\star\| = \|\boldsymbol{w}^\star\| \leq D$, along with with Eq. (13)) and Eq. (14) to get the desired bound. ■

In the remainder of this section, we present a few corollaries that are consequences of the theorem. The first corollary underscores that the rate of convergence in general is approximately $1/\varepsilon^2$, or, equivalently, $1/\sqrt{T}$.

Corollary 3 (Fixed step rate) *Assume that the conditions of Thm. 2 hold and that we run* FOBOS *for a predefined T iterations with $\eta_t = \frac{D}{\sqrt{7T}G}$ and that $(1 - cA\frac{D}{\sqrt{7T}G}) > 0$. Then*

$$\min_{t \in \{1,\ldots,T\}} f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t) \leq \frac{1}{T} \sum_{t=1}^{T} f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t) \leq \frac{3DG}{\sqrt{T}\left(1 - \frac{cAD}{G\sqrt{7T}}\right)} + \frac{f(\boldsymbol{w}^\star) + r(\boldsymbol{w}^\star)}{1 - \frac{cAD}{G\sqrt{7T}}}.$$

Proof Since we have $\eta_t = \eta$ for all t, the bound on the convergence rate from Thm. 2 becomes

$$\begin{aligned}
&\eta(1-cA\eta)T \min_{t\in\{1,\ldots,T\}} [f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t) - f(\boldsymbol{w}^\star) - r(\boldsymbol{w}^\star)] \\
&\leq \quad \eta(1-cA\eta) \sum_{t=1}^{T} [f(\boldsymbol{w}_t) - f(\boldsymbol{w}^\star)] + [r(\boldsymbol{w}_t) - r(\boldsymbol{w}^\star)] \;\leq\; D^2 + 7G^2 T\eta^2 \;\; .
\end{aligned}$$

Plugging in the specified value for η gives the bound. ■

Another direct consequence of Thm. 2 is convergence of the minimum over t when running FOBOS with $\eta_t \propto 1/\sqrt{t}$ or with non-summable step sizes decreasing to zero.

Corollary 4 (Convergence of decreasing step sizes) *Assume that the conditions of Thm. 2 hold and the step sizes η_t satisfy $\eta_t \to 0$ and $\sum_{t=1}^{\infty} \eta_t = \infty$. Then*

$$\liminf_{t\to\infty} f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t) - (f(\boldsymbol{w}^\star) + r(\boldsymbol{w}^\star)) = 0.$$

Finally, when f and r are Lipschitz with Lipschitz constant G, we immediately have

Corollary 5 *In addition to the conditions of Thm. 2, assume that the norm of any subgradient from ∂f and the norm of any subgradient from ∂r are bounded by G. Then*

$$\min_{t\in\{1,\ldots,T\}} (f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t)) - (f(\boldsymbol{w}^\star) + r(\boldsymbol{w}^\star)) \leq \frac{D^2 + 7G^2 \sum_{t=1}^{T} \eta_t^2}{2\sum_{t=1}^{T} \eta_t}. \tag{15}$$

Bounds of the above form, where we obtain convergence for one of the points in the sequence $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_T$ rather than the last point $\boldsymbol{w}_T$, are standard in subgradient optimization. The main reason that this weaker result occurs is due to the fact that we cannot guarantee a strict descent direction when using arbitrary subgradients (see, for example, Theorem 3.2.2 from Nesterov 2004). Another consequence of using non-differentiable functions means that analyses such as those carried out by Tseng (2000) and Chen and Rockafellar (1997) are difficult to apply, as the stronger rates rely on the existence and Lipschitz continuity of $\nabla f(\boldsymbol{w})$. However, it is possible to show linear convergence rates under suitable smoothness and strong convexity assumptions. When $\nabla f(\boldsymbol{w})$ is Lipschitz continuous, a more detailed analysis yields convergence rates of $1/\varepsilon$ (namely, $1/T$ in terms of number of iterations needed to be ε close to the optimum). A more complicated algorithm related to Nesterov's "estimate functions" (Nesterov, 2004) leads to $O(1/\sqrt{\varepsilon})$ convergence (Nesterov, 2007). For completeness, we give a simple proof of $1/T$ convergence in Appendix C. Finally, the above proof can be modified slightly to give convergence of the stochastic gradient method. In particular, we can replace $\boldsymbol{g}_t^f$ in the iterates of FOBOS with a stochastic estimate of the gradient $\tilde{\boldsymbol{g}}_t^f$, where $E[\tilde{\boldsymbol{g}}_t^f] \in \partial f(\boldsymbol{w}_t)$. We explore this approach in slightly more depth after performing a regret analysis for FOBOS below in Sec. 4 and describe stochastic convergence rates in Corollary 10.

We would like to make further comments on our proof of convergence for FOBOS and the assumptions underlying the proof. It is often not necessary to have a Lipschitz loss to guarantee boundedness of the subgradients of f and r, so in practice an assumption of bounded subgradients (as in Corollary 5 and in the sequel for online analysis) is not restrictive. If we know that an optimal $\boldsymbol{w}^\star$ lies in some compact set Ω and that $\Omega \subseteq \mathrm{dom}(f+r)$, then Theorem 24.7 of Rockafellar (1970) guarantees that ∂f and ∂r are bounded. The lingering question is thus whether we can guarantee that such a set Ω exists and that our iterates $\boldsymbol{w}_t$ remain in Ω. The following simple setting shows that ∂f and ∂r are indeed often bounded.

If $r(\boldsymbol{w})$ is a norm (possibly scaled) and f is lower bounded by 0, then we know that $r(\boldsymbol{w}^\star) \leq f(\boldsymbol{w}^\star) + r(\boldsymbol{w}^\star) \leq f(\boldsymbol{w}_1) + r(\boldsymbol{w}_1)$. Using standard bounds on norms, we get that for some $\gamma > 0$

$$\|\boldsymbol{w}^\star\|_\infty \leq \gamma r(\boldsymbol{w}^\star) \leq \gamma(f(\boldsymbol{w}_1) + r(\boldsymbol{w}_1)) = \gamma f(\boldsymbol{w}_1) \;\; ,$$

where for the last inequality we used the assumption that $r(\boldsymbol{w}_1) = 0$. Thus, we obtain that $\boldsymbol{w}^\star$ lies in a hypercube. We can easily project onto this box by truncating elements of $\boldsymbol{w}_t$ lying outside it at any iteration without affecting the bounds in Eq. (12) or Eq. (15). This additional Euclidean projection Π_Ω to an arbitrary convex set Ω with $\boldsymbol{w}^\star \in \Omega$ satisfies $\|\Pi_\Omega(\boldsymbol{w}_{t+1}) - \boldsymbol{w}^\star\| \leq \|\boldsymbol{w}_{t+1} - \boldsymbol{w}^\star\|$. Furthermore, so long as Ω is an ℓ_p-norm ball, we know that

$$r(\Pi_\Omega(\boldsymbol{w}_{t+1})) \leq r(\boldsymbol{w}_{t+1}) \ . \tag{16}$$

Thus, looking at Eq. (11), we notice that $r(\boldsymbol{w}^\star) - r(\boldsymbol{w}_{t+1}) \leq r(\boldsymbol{w}^\star) - r(\Pi_\Omega(\boldsymbol{w}_{t+1}))$ and the series of inequalities through Eq. (12) still hold (so long as $c\eta_{t+\frac{1}{2}}A \leq 1$, which it will if Ω is compact so that we can take $A = 0$). In general, so long as Eq. (16) holds and $\boldsymbol{w}^\star \in \Omega$, we can project $\boldsymbol{w}_{t+1}$ into Ω without affecting convergence guarantees. This property proves to be helpful in the regret analysis below.

4. Regret Analysis of FOBOS

In this section we provide regret analysis for FOBOS in online settings. In an online learning problem, we are given a sequence of functions $f_t : \mathbb{R}^n \to \mathbb{R}$. The learning goal is for the sequence of predictions $\boldsymbol{w}_t$ to attain low regret when compared to a single optimal predictor $\boldsymbol{w}^\star$. Formally, let $f_t(\boldsymbol{w})$ denote the loss suffered on the t^{th} input loss function when using a predictor $\boldsymbol{w}$. The regret of an online algorithm which uses $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t, \ldots$ as its predictors w.r.t. a fixed predictor $\boldsymbol{w}^\star$ while using a regularization function r is

$$R_{f+r}(T) = \sum_{t=1}^{T} \left[f_t(\boldsymbol{w}_t) + r(\boldsymbol{w}_t) - (f_t(\boldsymbol{w}^\star) + r(\boldsymbol{w}^\star))\right].$$

Ideally, we would like to achieve 0 regret to a stationary $\boldsymbol{w}^\star$ for arbitrary length sequences.

To achieve an online bound for a sequence of convex functions f_t, we modify arguments of Zinkevich (2003). Using the bound from Lemma 1, we can readily state and prove a theorem on the online regret of FOBOS. It is possible to avoid the boundedness assumptions in the proof of the theorem (getting a bound similar to that of Theorem 2 but for regret), however, we do not find it significantly more interesting. Aside from its reliance on Lemma 1, this proof is quite similar to Zinkevich's, so we defer it to Appendix A.

Theorem 6 *Assume that $\|\boldsymbol{w}_t - \boldsymbol{w}^\star\| \leq D$ for all iterations and the norm of the subgradient sets ∂f_t and ∂r are bounded above by G. Let $c > 0$ an arbitrary scalar. Then, the regret bound of* FOBOS *with $\eta_t = c/\sqrt{t}$ satisfies*

$$R_{f+r}(T) \leq 2GD + \left(\frac{D^2}{2c} + 7G^2c\right)\sqrt{T} \ .$$

The following Corollary is immediate from Theorem 6.

Corollary 7 *Assume the conditions of Theorem 6 hold. Then, setting $\eta_t = \frac{D}{4G\sqrt{t}}$, the regret of* FOBOS *is*

$$R_{f+r}(T) \leq 2GD + 4GD\sqrt{T} \ .$$

We can also obtain a better regret bound for FOBOS when the sequence of loss functions $f_t(\cdot)$ or the function $r(\cdot)$ is strongly convex. As demonstrated by Hazan et al. (2006), with the projected gradient method and strongly convex functions, it is possible to achieve regret on the order of $O(\log T)$ by using the curvature of the sequence of functions f_t rather than simply using convexity and linearity as in Theorems 2 and 6. We can extend these results to FOBOS for the case in which $f_t(\boldsymbol{w}) + r(\boldsymbol{w})$ is strongly convex, at least over the domain $\|\boldsymbol{w} - \boldsymbol{w}^\star\| \leq D$. For completeness, we recap a few definitions and provide the logarithmic regret bound for FOBOS. A function f is called H-strongly convex if

$$f(\boldsymbol{w}) \geq f(\boldsymbol{w}_t) + \langle \nabla f(\boldsymbol{w}_t), \boldsymbol{w} - \boldsymbol{w}_t \rangle + \frac{H}{2}\|\boldsymbol{w} - \boldsymbol{w}_t\|^2.$$

Thus, if r and the sequence of functions f_t are strongly convex with constants $H_f \geq 0$ and $H_r \geq 0$, we have $H = H_f + H_r$ and H-strong convexity gives

$$f_t(\boldsymbol{w}_t) - f(\boldsymbol{w}^\star) + r(\boldsymbol{w}_t) - r(\boldsymbol{w}^\star) \leq \left\langle \boldsymbol{g}_t^f + \boldsymbol{g}_t^r, \boldsymbol{w}_t - \boldsymbol{w}^\star \right\rangle - \frac{H}{2}\|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2 \ . \tag{17}$$

We do not need to assume that both f_t and r are strongly convex. We only need assume that at least one of them attains a positive strong convexity constant. For example, if $r(\boldsymbol{w}) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2$, then $H \geq \lambda$ so long as the functions f_t are convex. With Eq. (17) in mind, we can readily build on Hazan et al. (2006) and prove a stronger regret bound for the online learning case. The proof is similar to that of Hazan et al., so we defer it also to Appendix A.

Theorem 8 *Assume as in Theorem 6 that $\|\boldsymbol{w}_t - \boldsymbol{w}^\star\| \leq D$ and that ∂f_t and ∂r are bounded above by G. Assume further that $f_t + r$ is H-strongly convex for all t. Then, when using step sizes $\eta_t = 1/Ht$, the regret of* FOBOS *is*

$$R_{f+r}(T) = O\left(\frac{G^2}{H}\log T\right).$$

We now provide an easy lemma showing that for Lipschitz losses with ℓ_2^2 regularization, the boundedness assumptions above hold. This, for example, includes the interesting case of support vector machines. The proof is not difficult but relies tacitly on a later result, so we leave it to Appendix A.

Lemma 9 *Let the functions f_t be G-Lipschitz so that $\|\partial f_t(\boldsymbol{w})\| \leq G$. Let $r(\boldsymbol{w}) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2$. Then $\|\boldsymbol{w}^\star\| \leq G/\lambda$ and the iterates $\boldsymbol{w}_t$ generated by* FOBOS *satisfy $\|\boldsymbol{w}_t\| \leq G/\lambda$.*

Using the regret analysis for online learning, we are able to return to learning in a batch setting and give stochastic convergence rates for FOBOS. We build on results of Shalev-Shwartz et al. (2007) and assume as in Sec. 3 that we are minimizing $f(\boldsymbol{w}) + r(\boldsymbol{w})$. Indeed, suppose that on each step of FOBOS, we choose instead of some $\boldsymbol{g}_t^f \in \partial f(\boldsymbol{w}_t)$ a stochastic estimate of the gradient $\tilde{\boldsymbol{g}}_t^f$ where $E[\tilde{\boldsymbol{g}}_t^f] \in \partial f(\boldsymbol{w}_t)$. We assume that we still use the true r (which is generally easy, as it is simply the regularization function). It is straightforward to use theorems 6 and 8 above as in the derivation of theorems 2 and 3 from Shalev-Shwartz et al. (2007) to derive the following corollary on the expected convergence rate of FOBOS as well as a guarantee of convergence with high probability.

Corollary 10 *Assume that the conditions on ∂f, ∂r, and $\boldsymbol{w}^\star$ hold as in the previous theorems and let* FOBOS *be run for T iterations. Let s be an integer chosen uniformly at random from $\{1, \ldots, T\}$. If $\eta_t = \frac{D}{4G\sqrt{t}}$, then*

$$E_s[f(\boldsymbol{w}_s) + r(\boldsymbol{w}_s)] \leq f(\boldsymbol{w}^\star) + r(\boldsymbol{w}^\star) + \frac{2GD + 4GD\sqrt{T}}{T}.$$

With probability at least $1 - \delta$,

$$f(\boldsymbol{w}_s) + r(\boldsymbol{w}_s) \leq f(\boldsymbol{w}^\star) + r(\boldsymbol{w}^\star) + \frac{2GD + 4GD\sqrt{T}}{\delta T}.$$

If $f + r$ is H-strongly convex and we choose $\eta_t \propto 1/t$, we have

$$E_s[f(\boldsymbol{w}_s) + r(\boldsymbol{w}_s)] = f(\boldsymbol{w}^\star) + r(\boldsymbol{w}^\star) + O\left(\frac{G^2 \log T}{HT}\right)$$

and with probability at least $1 - \delta$,

$$f(\boldsymbol{w}_s) + r(\boldsymbol{w}_s) = f(\boldsymbol{w}^\star) + r(\boldsymbol{w}^\star) + O\left(\frac{G^2 \log T}{H\delta T}\right).$$

5. Derived Algorithms

In this section we derive a few variants of FOBOS by considering different regularization functions. The emphasis of the section is on non-differentiable regularization functions, such as the ℓ_1 norm of $\boldsymbol{w}$, which lead to sparse solutions. We also derive simple extensions to mixed-norm regularization (Zhao et al., 2006) that build on the first part of this section.

First, we make a few changes to notation. To simplify our derivations, we denote by $\boldsymbol{v}$ the vector $\boldsymbol{w}_{t+\frac{1}{2}} = \boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f$ and let $\tilde{\lambda}$ denote $\eta_{t+\frac{1}{2}} \cdot \lambda$. Using this newly introduced notation the problem given in Eq. (3) can be rewritten as

$$\operatorname*{minimize}_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}-\boldsymbol{v}\|^2 + \tilde{\lambda} r(\boldsymbol{w}). \tag{18}$$

Let $[z]_+$ denote $\max\{0,z\}$. For completeness, we provide full derivations for all the regularization functions we consider, but for brevity we do not state formally well established technical lemmas. We note that many of the following results were given tacitly by Wright et al. (2009).

5.1 FOBOS with ℓ_1 Regularization

The update obtained by choosing $r(\boldsymbol{w}) = \lambda\|\boldsymbol{w}\|_1$ is simple and intuitive. First note that the objective is decomposable as we can rewrite Eq. (18) as

$$\operatorname*{minimize}_{\boldsymbol{w}} \sum_{j=1}^{n} \left(\frac{1}{2}(w_j - v_j)^2 + \tilde{\lambda}|w_j|\right).$$

Let us focus on a single coordinate of $\boldsymbol{w}$ and for brevity omit the index j. Let $w^\star$ denote the minimizer of $\frac{1}{2}(w-v)^2 + \tilde{\lambda}|w|$. Clearly, $w^\star \cdot v \geq 0$. If it were not, then we would have $w^\star \cdot v < 0$, however $\frac{1}{2}v^2 < \frac{1}{2}v^2 - w^\star \cdot v + \frac{1}{2}(w^\star)^2 < \frac{1}{2}(v-w^\star)^2 + \tilde{\lambda}|w^\star|$, contradicting the supposed optimality of $w^\star$. The symmetry of the objective in v also shows us that we can assume that $v \geq 0$; we therefore need to minimize $\frac{1}{2}(w-v)^2 + \tilde{\lambda}w$ subject to the constraint that $w \geq 0$. Introducing a Lagrange multiplier $\beta \geq 0$ for the constraint, we have the Lagrangian $\frac{1}{2}(w-v)^2 + \tilde{\lambda}w - \beta w$. By taking the derivative of the Langrangian with respect to w and setting the result to zero, we get that the optimal solution is $w^\star = v - \tilde{\lambda} + \beta$. If $w^\star > 0$, then from the complimentary slackness condition that the optimal pair of $w^\star$ and β must have $w^\star\beta = 0$ (Boyd and Vandenberghe, 2004) we must have $\beta = 0$, and therefore $w^\star = v - \tilde{\lambda}$. If $v < \tilde{\lambda}$, then $v - \tilde{\lambda} < 0$, so we must have $\beta > 0$ and again by complimentary slackness, $w^\star = 0$. The case when $v \leq 0$ is analogous and amounts to simply flipping signs. Summarizing and expanding notation, the components of the optimal solution $\boldsymbol{w}^\star = \boldsymbol{w}_{t+1}$ are computed from $\boldsymbol{w}_{t+\frac{1}{2}}$ as

$$w_{t+1,j} = \operatorname{sign}\left(w_{t+\frac{1}{2},j}\right)\left[|w_{t+\frac{1}{2},j}| - \tilde{\lambda}\right]_+ = \operatorname{sign}\left(w_{t,j} - \eta_t g_{t,j}^f\right)\left[\left|w_{t,j} - \eta_t g_{t,j}^f\right| - \eta_{t+\frac{1}{2}} \cdot \lambda\right]_+. \tag{19}$$

Note that this update can lead to sparse solutions. Whenever the absolute value of a component of $\boldsymbol{w}_{t+\frac{1}{2}}$ is smaller than $\tilde{\lambda}$, the corresponding component in $\boldsymbol{w}_{t+1}$ is set to zero. Thus, Eq. (19) gives a simple online and offline method for minimizing a convex f with ℓ_1 regularization.

Such soft-thresholding operations are common in the statistics literature and have been used for some time (Donoho, 1995; Daubechies et al., 2004). Langford et al. (2008) recently proposed and analyzed the same update, terming it the "truncated gradient." The analysis presented here is different from the analysis in the aforementioned paper as it stems from a more general framework. Indeed, as illustrated in this section, the derivation and method is also applicable to a wide variety of regularization functions. Nevertheless, both analyses merit consideration as they shed light from different angles on the problem of learning sparse models using gradients, stochastic gradients, or online methods. This update can also be implemented very efficiently when the support of $\boldsymbol{g}_t^f$ is small (Langford et al., 2008), but we defer details to Sec. 6, where we give a unified view that facilitates efficient implementation for all the norm regularization functions we discuss.

5.2 FOBOS with ℓ_2^2 Regularization

When $r(\boldsymbol{w}) = \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$, we obtain a very simple optimization problem,

$$\underset{\boldsymbol{w}}{\text{minimize}}\ \frac{1}{2}\|\boldsymbol{w}-\boldsymbol{v}\|^2 + \frac{1}{2}\tilde{\lambda}\|\boldsymbol{w}\|^2,$$

where for conciseness of the solution we replace $\tilde{\lambda}$ with $\frac{1}{2}\tilde{\lambda}$. Differentiating the above objective and setting the result equal to zero, we have $\boldsymbol{w}^\star - \boldsymbol{v} + \tilde{\lambda}\boldsymbol{w}^\star = 0$, which, using the original notation, yields the update

$$\boldsymbol{w}_{t+1} = \frac{\boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f}{1+\tilde{\lambda}}. \tag{20}$$

Informally, the update simply shrinks $\boldsymbol{w}_{t+1}$ back toward the origin after each gradient-descent step. In Sec. 7 we briefly compare the resulting FOBOS update to modern stochastic gradient techniques and show that the FOBOS update exhibits similar empirical behavior.

5.3 FOBOS with ℓ_2 Regularization

A lesser used regularization function is the ℓ_2 norm of the weight vector. By setting $r(\boldsymbol{w}) = \tilde{\lambda}\|\boldsymbol{w}\|$ we obtain the following problem,

$$\underset{\boldsymbol{w}}{\text{minimize}}\ \frac{1}{2}\|\boldsymbol{w}-\boldsymbol{v}\|^2 + \tilde{\lambda}\|\boldsymbol{w}\|. \tag{21}$$

The solution of Eq. (21) must be in the direction of $\boldsymbol{v}$ and takes the form $\boldsymbol{w}^\star = s\boldsymbol{v}$ where $s \geq 0$. To show that this is indeed the form of the solution, let us assume for the sake of contradiction that $\boldsymbol{w}^\star = s\boldsymbol{v} + \boldsymbol{u}$ where $\boldsymbol{u}$ is in the null space of $\boldsymbol{v}$ (if $\boldsymbol{u}$ has any components parallel to $\boldsymbol{v}$, we can add those to $s\boldsymbol{v}$ and obtain an orthogonal $\boldsymbol{u}'$) and s may be negative. Since $\boldsymbol{u}$ is orthogonal to $\boldsymbol{v}$, the objective function can be expressed in terms of s, $\boldsymbol{v}$, and $\boldsymbol{u}$ as

$$\frac{1}{2}(1-s)^2\|\boldsymbol{v}\|^2 + \frac{1}{2}\|\boldsymbol{u}\|^2 + \tilde{\lambda}(\|\boldsymbol{v}\| + \|\boldsymbol{u}\|) \geq \frac{1}{2}(1-s)^2\|\boldsymbol{v}\|^2 + \tilde{\lambda}\|\boldsymbol{v}\|.$$

Thus, $\boldsymbol{u}$ must be equal to the zero vector, $\boldsymbol{u} = \boldsymbol{0}$, and we can write the optimization problem as

$$\underset{s}{\text{minimize}}\ \frac{1}{2}(1-s)^2\|\boldsymbol{v}\|^2 + \tilde{\lambda}s\|\boldsymbol{v}\| \ .$$

Next note that a negative value for s cannot constitute the optimal solution. Indeed, if $s < 0$, then

$$\frac{1}{2}(1-s)^2\|\boldsymbol{v}\|^2 + \tilde{\lambda}s\|\boldsymbol{v}\| \ < \ \frac{1}{2}\|\boldsymbol{v}\|^2 \ .$$

This implies that by setting $s = 0$ we can obtain a lower objective function, and this precludes a negative value for s as an optimal solution. We therefore end again with a constrained scalar optimization problem, $\text{minimize}_{s\geq 0}\ \frac{1}{2}(1-s)^2\|\boldsymbol{v}\|^2 + \tilde{\lambda}s\|\boldsymbol{v}\|$. The Lagrangian of this problem is

$$\frac{1}{2}(1-s)^2\|\boldsymbol{v}\|^2 + \tilde{\lambda}s\|\boldsymbol{v}\| - \beta s \ ,$$

where $\beta \geq 0$. By taking the derivative of the Langrangian with respect to s and setting the result to zero, we get that $(s-1)\|\boldsymbol{v}\|^2 + \tilde{\lambda}\|\boldsymbol{v}\| - \beta = 0$ which gives the following closed form solution: $s = 1 - \tilde{\lambda}/\|\boldsymbol{v}\| + \beta/\|\boldsymbol{v}\|^2$. Whenever $s > 0$ then the complimentary slackness conditions imply that $\beta = 0$ and s can be further simplified and written as $s = 1 - \tilde{\lambda}/\|\boldsymbol{v}\|$. The last expression is positive iff $\|\boldsymbol{v}\| > \tilde{\lambda}$. If $\|\boldsymbol{v}\| < \tilde{\lambda}$, then β must be positive and complimentary slackness implies that $s = 0$.

Summarizing, the second step of the FOBOS update with ℓ_2 regularization amounts to

$$\boldsymbol{w}_{t+1} = \left[1 - \frac{\tilde{\lambda}}{\|\boldsymbol{w}_{t+\frac{1}{2}}\|}\right]_+ \boldsymbol{w}_{t+\frac{1}{2}} \ = \ \left[1 - \frac{\tilde{\lambda}}{\|\boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f\|}\right]_+ (\boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f) \ .$$

Thus, the ℓ_2 regularization results in a zero weight vector under the condition that $\|\boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f\| \leq \tilde{\lambda}$. This condition is rather more stringent for sparsity than the condition for ℓ_1 (where a weight is sparse based only on its value, while here, sparsity happens only if the entire weight vector has ℓ_2-norm less than $\tilde{\lambda}$), so it is unlikely to hold in high dimensions. However, it does constitute a very important building block when using a mixed ℓ_1/ℓ_2-norm as the regularization function, as we show in the sequel.

5.4 FOBOS with ℓ_∞ Regularization

We now turn to a less explored regularization function, the ℓ_∞ norm of $\boldsymbol{w}$. This form of regularization is not capable of providing strong guarantees against over-fitting on its own as many of the weights of $\boldsymbol{w}$ may not be penalized. However, there are settings in which it is desirable to consider blocks of variables as a group, such as ℓ_1/ℓ_∞ regularization. We continue to defer the discussion on mixing different norms and focus merely on the ℓ_∞ norm as it serves as a building block. That is, we are interested in obtaining an efficient solution to the following problem,

$$\operatorname*{minimize}_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{v}\|^2 + \tilde{\lambda}\|\boldsymbol{w}\|_\infty \quad . \tag{22}$$

It is possible to derive an efficient algorithm for finding the minimizer of Eq. (22) using properties of the subgradient set of $\|\boldsymbol{w}\|_\infty$. However, a solution to the dual form of Eq. (22) is well established. Recalling that the conjugate of the quadratic function is a quadratic function and the conjugate of the ℓ_∞ norm is the ℓ_1 barrier function, we immediately obtain that the dual of the problem given by Eq. (22) is

$$\operatorname*{maximize}_{\boldsymbol{\alpha}} -\frac{1}{2}\|\boldsymbol{\alpha} - \boldsymbol{v}\|_2^2 \;\; \text{s.t.} \;\; \|\boldsymbol{\alpha}\|_1 \leq \tilde{\lambda} \;\; . \tag{23}$$

Moreover, the vector of dual variables $\boldsymbol{\alpha}$ satisfies the relation $\boldsymbol{\alpha} = \boldsymbol{v} - \boldsymbol{w}$. Thus, by solving the dual form in Eq. (23) we can readily obtain a solution for Eq. (22). The problem defined by Eq. (23) is equivalent to performing Euclidean projection onto the ℓ_1 ball and has been studied by numerous authors. The solution that we overview here is based on recent work of Duchi et al. (2008). The maximizer of Eq. (23), denoted $\boldsymbol{\alpha}^\star$, is of the form

$$\alpha_j^\star = \operatorname{sign}(v_j)\left[|v_j| - \theta\right]_+ \quad , \tag{24}$$

where θ is a non-negative scalar. Duchi et al. (2008) describe a linear time algorithm for finding θ. We thus skip the analysis of the algorithm and focus on its core properties that affect the solution of the original problem Eq. (22). To find θ we need to locate a pivot element in $\boldsymbol{v}$, denoted by the ρ^{th} order statistic $v_{(\rho)}$ (where $v_{(1)}$ is the largest magnitude entry of $\boldsymbol{v}$), with the following property, $v_{(\rho)}$ is the smallest magnitude element in $\boldsymbol{v}$ such that

$$\sum_{j:|v_j|>|v_{(\rho)}|} \left(|v_j| - |v_{(\rho)}|\right) < \tilde{\lambda} \;\; .$$

If all the elements in $\boldsymbol{v}$ (assuming that we have added an extra 0 element to handle the smallest entry of $\boldsymbol{v}$) satisfy the above requirement then the optimal choice for θ is 0. Otherwise,

$$\theta = \frac{1}{\rho}\left(\sum_{j:|v_j|>|v_{(\rho)}|} |v_j| - \tilde{\lambda}\right) \quad .$$

Thus, the optimal choice of θ is zero when $\sum_{j=1}^n |v_j| \leq \tilde{\lambda}$ (this is, not coincidentally, simply the subgradient condition for optimality of the zero vector in Eq. (22)).

Using the linear relationship $\boldsymbol{\alpha} = \boldsymbol{v} - \boldsymbol{w} \Rightarrow \boldsymbol{w} = \boldsymbol{v} - \boldsymbol{\alpha}$ along with the solution of the dual problem as given by Eq. (24), we obtain the following solution for Eq. (22),

$$w_{t+1,j} = \operatorname{sign}\left(w_{t+\frac{1}{2},j}\right)\min\left\{\left|w_{t+\frac{1}{2},j}\right| \, , \, \theta\right\} \quad . \tag{25}$$

As stated above, $\theta = 0$ iff $\|\boldsymbol{w}_{t+\frac{1}{2}}\|_1 \leq \tilde{\lambda}$ and otherwise $\theta > 0$ and can be found in $O(n)$ steps. In words, the components of the vector $\boldsymbol{w}_{t+1}$ are the result of capping all of the components of $\boldsymbol{w}_t$ at θ where θ is zero when the 1-norm of $\boldsymbol{w}_{t+\frac{1}{2}}$ is smaller than $\tilde{\lambda}$. Interestingly, this property shares a duality resemblance with the ℓ_2-regularized update, which results in a zero weight vector when the 2-norm (which is self-dual) of $\boldsymbol{v}$ is less than $\tilde{\lambda}$. We can exploit these properties in the context of mixed-norm regularization to achieve sparse solutions for complex prediction problems, which we describe in the sequel and for which we present preliminary results in Sec. 7. Before doing so, we present one more norm-related regularization.

5.5 FOBOS with Berhu Regularization:

We now consider a regularization function which is a hybrid of the ℓ_1 and ℓ_2 norms. Similar to ℓ_1 regularization, Berhu (for inverse Huber) regularization results in sparse solutions, but its hybridization with ℓ_2^2 regularization prevents the weights from being excessively large. Berhu regularization (Owen, 2006) is defined as

$$r(\boldsymbol{w}) = \lambda \sum_{j=1}^{n} b(w_j) = \lambda \sum_{j=1}^{n} \left[|w_j| \left[\!\left[|w_j| \leq \gamma \right]\!\right] + \frac{w_j^2 + \gamma^2}{2\gamma} \left[\!\left[|w_j| > \gamma \right]\!\right] \right].$$

In the above, $[\![\cdot]\!]$ is 1 if its argument is true and is 0 otherwise. The positive scalar γ controls the value for which the Berhu regularization switches from ℓ_1 mode to ℓ_2 mode. Formally, when $w_j \in [-\gamma, \gamma]$, $r(\boldsymbol{w})$ behaves as ℓ_1, and for w_j outside this region, the Berhu penalty behaves as ℓ_2^2. The Berhu penalty is convex and differentiable except at 0, where it has the same subdifferential set as $\lambda \|\boldsymbol{w}\|_1$.

To find a closed form update from $\boldsymbol{w}_{t+\frac{1}{2}}$ to $\boldsymbol{w}_{t+1}$, we minimize in each variable $\frac{1}{2}(w-v) + \tilde{\lambda} b(w)$; the derivation is fairly standard but technical and is provided in Appendix B. The end result is aesthetic and captures the ℓ_1 and ℓ_2 regions of the Berhu penalty,

$$w_{t+1,j} = \begin{cases} 0 & \left| w_{t+\frac{1}{2},j} \right| \leq \tilde{\lambda} \\ \operatorname{sign}(w_{t+\frac{1}{2},j}) \left[|w_{t+\frac{1}{2},j}| - \tilde{\lambda} \right] & \tilde{\lambda} < |w_{t+\frac{1}{2},j}| \leq \tilde{\lambda} + \gamma \\ \frac{w_{t+\frac{1}{2},j}}{1 + \tilde{\lambda}/\gamma} & \gamma + \tilde{\lambda} < \left| w_{t+\frac{1}{2},j} \right| \end{cases} . \tag{26}$$

Indeed, as Eq. (26) indicates, the update takes one of two forms, depending on the magnitude of the coordinate of $w_{t+\frac{1}{2},j}$. If $|w_{t+\frac{1}{2},j}|$ is greater than $\gamma + \tilde{\lambda}$, the update is identical to the update for ℓ_2^2-regularization of Eq. (20), while if the value is no larger than $\gamma + \tilde{\lambda}$, the resulting update is equivalent to the update for ℓ_1-regularization of Eq. (19).

5.6 Extension to Mixed Norms

We saw above that when using either the ℓ_2 or the ℓ_∞ norm as the regularization function, we obtain an all zeros vector if $||\boldsymbol{w}_{t+\frac{1}{2}}||_2 \leq \tilde{\lambda}$ or $||\boldsymbol{w}_{t+\frac{1}{2}}||_1 \leq \tilde{\lambda}$, respectively. The zero vector does not carry any generalization properties, which surfaces a concern regarding the usability of the these norms as a form of regularization. This seemingly problematic phenomenon can, however, be useful in the setting we discuss now. In many applications, the set of weights can be grouped into subsets where each subset of weights should be dealt with uniformly. For example, in multiclass categorization problems each class r may be associated with a different weight vector $\boldsymbol{w}^r$. The prediction for an instance $\boldsymbol{x}$ is a vector $\langle \boldsymbol{w}^1, \boldsymbol{x} \rangle, \ldots, \langle \boldsymbol{w}^k, \boldsymbol{x} \rangle$ where k is the number of different classes. The predicted class is the index of the inner-product attaining the largest of the k values, $\operatorname{argmax}_j \langle \boldsymbol{w}^j, \boldsymbol{x} \rangle$. Since all the weight vectors operate over the same instance space, in order to achieve a sparse solution, it may be beneficial to tie the weights corresponding to the same input feature. That is, we would like to employ a regularization function that tends to zero the row of weights $w_j^1, \ldots, w_j^k$ *simultaneously*. In these circumstances, the nullification of the entire weight vector by ℓ_2 and ℓ_∞ regularization becomes a powerful tool.

Formally, let W represent a $n \times k$ matrix where the j^{th} column of the matrix is the weight vector $\boldsymbol{w}^j$ associated with class j. Thus, the i^{th} row corresponds to the weight of the i^{th} feature with respect to all classes. The mixed ℓ_r/ℓ_s-norm (Obozinski et al., 2007) of W, denoted $\|W\|_{\ell_r/\ell_s}$, is obtained by computing the ℓ_s-norm of each row of W and then applying the ℓ_r-norm to the resulting n dimensional vector, for instance, $\|W\|_{\ell_1/\ell_\infty} = \sum_{j=1}^{n} \max_j |W_{i,j}|$. Thus, in a mixed-norm regularized optimization problem (such as multiclass learning), we seek the minimizer of the objective function

$$f(W) + \lambda \|W\|_{\ell_r/\ell_s} .$$

Given the specific variants of various norms described above, the FOBOS update for the ℓ_1/ℓ_∞ and the ℓ_1/ℓ_2 mixed-norms is readily available. Let $\bar{\boldsymbol{w}}^r$ denote the r^{th} row of W. Analogously to the standard norm-based regularization, we let $V = W_{t+\frac{1}{2}}$ be a shorthand for the result of the gradient step. For the ℓ_1, ℓ_p mixed-norm, where $p = 2$ or $p = \infty$, we need to solve the problem

$$\operatorname*{minimize}_{W} \frac{1}{2}\|W - V\|_{\mathrm{Fr}}^2 + \tilde{\lambda}\|W\|_{\ell_1/\ell_2} \equiv \operatorname*{minimize}_{\bar{\boldsymbol{w}}^1,\ldots,\bar{\boldsymbol{w}}^k} \sum_{i=1}^{n} \left(\frac{1}{2}\left\|\bar{\boldsymbol{w}}^i - \bar{\boldsymbol{v}}^i\right\|_2^2 + \tilde{\lambda}\left\|\bar{\boldsymbol{w}}^i\right\|_p \right) , \tag{27}$$

where $\bar{\boldsymbol{v}}^i$ is the i^{th} row of V. It is immediate to see that the problem given in Eq. (27) is decomposable into n separate problems of dimension k, each of which can be solved by the procedures described in the prequel. The end result of solving these types of mixed-norm problems is a sparse matrix with numerous zero rows. We demonstrate the merits of FOBOS with mixed-norms in Sec. 7.

6. Efficient Implementation in High Dimensions

In many settings, especially online learning, the weight vector $\boldsymbol{w}_t$ and the gradients $\boldsymbol{g}_t^f$ reside in a very high-dimensional space, but only a relatively small number of the components of $\boldsymbol{g}_t^f$ are non-zero. Such settings are prevalent, for instance, in text-based applications: in text categorization, the full dimension corresponds to the dictionary or set of tokens that is being employed while each gradient is typically computed from a single or a few documents, each of which contains words and bigrams constituting only a small subset of the full dictionary. The need to cope with gradient sparsity becomes further pronounced in mixed-norm problems, as a single component of the gradient may correspond to an entire row of W. Updating the entire matrix because a few entries of $\boldsymbol{g}_t^f$ are non-zero is clearly undesirable. Thus, we would like to extend our methods to cope efficiently with gradient sparsity. For concreteness, we focus in this section on the efficient implementation of ℓ_1, ℓ_2, and ℓ_∞ regularization, recognizing that the extension to mixed-norms (as in the previous section) is straightforward. The upshot of following proposition is that when $\boldsymbol{g}_t^f$ is sparse, we can use lazy evaluation of the weight vectors and defer to later rounds the update of components of $\boldsymbol{w}_t$ whose corresponding gradient entries are zero. We detail this after the proposition, which is technical so the interested reader may skip the proof to see the simple algorithms for lazy evaluation.

Proposition 11 *Let $\boldsymbol{w}_T$ be the end result of solving a succession of T self-similar optimization problems for $t = 1, \ldots, T$,*

$$\mathcal{P}.1: \quad \boldsymbol{w}_t = \operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_{t-1}\|^2 + \lambda_t \|\boldsymbol{w}\|_q \ . \tag{28}$$

Let $\boldsymbol{w}^\star$ be the optimal solution of the following optimization problem:

$$\mathcal{P}.2: \quad \boldsymbol{w}^\star = \operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_0\|^2 + \left(\sum_{t=1}^{T} \lambda_t \right) \|\boldsymbol{w}\|_q \ .$$

Then for $q \in \{1, 2, \infty\}$ the vectors $\boldsymbol{w}_T$ and $\boldsymbol{w}^\star$ are identical.

Proof It suffices to show that the proposition is correct for $T = 2$ and then use an inductive argument, because the proposition trivially holds for $T = 1$. We provide here a direct proof for each norm separately by examining the updates we derived in Sec. 5 and showing that $\boldsymbol{w}_2 = \boldsymbol{w}^\star$.

Note that the objective functions are separable for $q = 1$. Therefore, for ℓ_1-regularization it suffices to prove the proposition for any component of the vector $\boldsymbol{w}$. We omit the index of the component and denote by $w_0, w_1, w_2, w_3, \ldots$ one coordinate of $\boldsymbol{w}$ along the iterations of $\mathcal{P}.1$ and by $w^\star$ the result for the same component when solving $\mathcal{P}.2$. We need to show that $w^\star = w_2$. Expanding the ℓ_1-update of Eq. (19) over two iterations we get the following,

$$w_2 = \text{sign}(w_1)\,[|w_1| - \lambda_2]_+ = \text{sign}(w_1)\left[\left|\text{sign}(w_0)\,[|w_0| - \lambda_1]_+\right| - \lambda_2\right]_+ = \text{sign}(w_0)\,[|w_0| - \lambda_1 - \lambda_2]_+ \quad ,$$

where we used the positivity of $|\cdot|$. Examining $\mathcal{P}.2$ and using Eq. (19) again we get

$$w^\star = \text{sign}(w_0)\,[|w_0| - \lambda_1 - \lambda_2]_+ \quad .$$

Therefore, $w^\star = w_2$ as claimed.

Next we prove the proposition for ℓ_2, returning to using the entire vector for the proof. Using the explicit ℓ_2-update from Eq. (20), we can expand the norm of the vector $\boldsymbol{w}_1$ due to the program $\mathcal{P}.1$ as follows,

$$\|\boldsymbol{w}_1\| \;=\; \left[1 - \frac{\lambda_1}{\|\boldsymbol{w}_0\|}\right]_+ \|\boldsymbol{w}_0\| \;=\; [\|\boldsymbol{w}_0\| - \lambda_1]_+ \quad .$$

Similarly, we get that $\|\boldsymbol{w}_2\| = [\|\boldsymbol{w}_1\| - \lambda_2]_+$. Combining the norm equalities we see that the norm of $\boldsymbol{w}_2$ due to the succession of the two updates is

$$\|\boldsymbol{w}_2\| \;=\; \left[[\|\boldsymbol{w}_0\| - \lambda_1]_+ - \lambda_2\right]_+ \;=\; [\|\boldsymbol{w}_0\| - \lambda_1 - \lambda_2]_+ \quad .$$

Computing directly the norm of $\boldsymbol{w}^\star$ due to the update given by Eq. (20) yields

$$\|\boldsymbol{w}^\star\| \;=\; \left[1 - \frac{\lambda_1 + \lambda_2}{\|\boldsymbol{w}_0\|}\right]_+ \|\boldsymbol{w}_0\| \;=\; [\|\boldsymbol{w}_0\| - \lambda_1 - \lambda_2]_+ \quad .$$

Thus, $\boldsymbol{w}^\star$ and $\boldsymbol{w}_2$ have the same norm. Since the update itself retains the direction of the original vector $\boldsymbol{w}_0$, we get that $\boldsymbol{w}^\star = \boldsymbol{w}_2$ as needed.

We now turn to the most complicated update and proof of the three norms, the ℓ_∞ norm. We start by recapping the programs $\mathcal{P}.1$ and $\mathcal{P}.2$ for $T = 2$ and $q = \infty$,

$$\mathcal{P}.1: \quad \boldsymbol{w}_1 \;=\; \underset{\boldsymbol{w}}{\text{argmin}} \left\{ \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_0\|^2 + \lambda_1 \|\boldsymbol{w}\|_\infty \right\} \tag{29}$$

$$\boldsymbol{w}_2 \;=\; \underset{\boldsymbol{w}}{\text{argmin}} \left\{ \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_1\|^2 + \lambda_2 \|\boldsymbol{w}\|_\infty \right\} \quad , \tag{30}$$

$$\mathcal{P}.2: \quad \boldsymbol{w}^\star \;=\; \underset{\boldsymbol{w}}{\text{argmin}} \left\{ \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{v}\|_2^2 + (\lambda_1 + \lambda_2) \|\boldsymbol{w}\|_\infty \right\} \quad . \tag{31}$$

We prove the equivalence of the two programs in two stages. First, we examine the case $\|\boldsymbol{w}_0\|_1 > \lambda_1 + \lambda_2$, and then consider the complement case $\|\boldsymbol{w}_0\|_1 \leq \lambda_1 + \lambda_2$. For concreteness and simplicity, we assume that $\boldsymbol{w}_0 \succeq \boldsymbol{0}$, since, clearly, the objective is symmetric in $\boldsymbol{w}_0$ and $-\boldsymbol{w}_0$. We thus can assume that all entries of $\boldsymbol{w}_0$ are non-negative. In the proof we use the following operators: $[\boldsymbol{v}]_+$ now denotes the positive component of each entry of $\boldsymbol{v}$, $\min\{\boldsymbol{v}, \theta\}$ denotes the component-wise minimum between the elements of $\boldsymbol{v}$ and θ, and likewise $\max\{\boldsymbol{v}, \theta\}$ is the component-wise maximum. Starting with the case $\|\boldsymbol{w}_0\|_1 > \lambda_1 + \lambda_2$, we examine Eq. (29). From Lagrange duality we know that that $\boldsymbol{w}_1 = \boldsymbol{w}_0 - \boldsymbol{\alpha}_1$, where $\boldsymbol{\alpha}_1$ is the solution of

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \frac{1}{2}\|\boldsymbol{\alpha} - \boldsymbol{w}_0\|_2^2 \;\text{ s.t. }\; \|\boldsymbol{\alpha}\|_1 \leq \lambda_1 \quad .$$

As described by Duchi et al. (2008) and reviewed above in Sec. 5, $\boldsymbol{\alpha}_1 = [\boldsymbol{w}_0 - \theta_1]_+$ for some $\theta_1 \in \mathbb{R}_+$. The form of $\boldsymbol{\alpha}_1$ readily translates to the following form for $\boldsymbol{w}_1$: $\boldsymbol{w}_1 = \boldsymbol{w}_0 - \boldsymbol{\alpha}_1 = \min(\boldsymbol{w}_0, \theta_1)$. Applying similar reasoning to the second step of $\mathcal{P}.1$ yields $\boldsymbol{w}_2 = \boldsymbol{w}_1 - \boldsymbol{\alpha}_2 = \boldsymbol{w}_0 - \boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2$, where $\boldsymbol{\alpha}_2$ is the minimizer of

$$\frac{1}{2}\|\boldsymbol{\alpha} - \boldsymbol{w}_1\|_2^2 = \frac{1}{2}\|\boldsymbol{\alpha} - (\boldsymbol{w}_0 - \boldsymbol{\alpha}_1)\|_2^2 \quad \text{s.t. } \|\boldsymbol{\alpha}\|_1 \leq \lambda_2 \quad .$$

Again, we have $\boldsymbol{\alpha}_2 = [\boldsymbol{w}_1 - \theta_2]_+ = [\boldsymbol{w}_0 - \boldsymbol{\alpha}_1 - \theta_2]_+$ for some $\theta_2 \in \mathbb{R}_+$. The successive steps then imply that

$$\boldsymbol{w}_2 = \min\{\boldsymbol{w}_1, \theta_2\} = \min\left\{\min\{\boldsymbol{w}_0, \theta_1\}, \theta_2\right\} \ .$$

We next show that regardless of the ℓ_1-norm of $\boldsymbol{w}_0$, $\theta_2 \leq \theta_1$. Intuitively, if $\theta_2 > \theta_1$, the second minimization step of $\mathcal{P}.1$ would perform no shrinkage of $\boldsymbol{w}_1$ to get $\boldsymbol{w}_2$. Formally, assume for the sake of contradiction that $\theta_2 > \theta_1$. Under this assumption, we would have that $\boldsymbol{w}_2 = \min\{\min\{\boldsymbol{w}_0, \theta_1\}, \theta_2\} = \min\{\boldsymbol{w}_0, \theta_1\} = \boldsymbol{w}_1$. In turn, we obtain that $\mathbf{0}$ belongs to the subgradient set of Eq. (30) when evaluated at $\boldsymbol{w} = \boldsymbol{w}_1$, thus,

$$\mathbf{0} \in \boldsymbol{w}_1 - \boldsymbol{w}_1 + \lambda_2 \partial \left\|\boldsymbol{w}_1\right\|_\infty = \lambda_2 \partial \left\|\boldsymbol{w}_1\right\|_\infty \ .$$

Clearly, the set $\partial \left\|\boldsymbol{w}_1\right\|_\infty$ can contain $\mathbf{0}$ only when $\boldsymbol{w}_1 = \mathbf{0}$. Since we assumed that $\lambda_1 < \left\|\boldsymbol{w}_0\right\|_1$, and hence that $\boldsymbol{\alpha}_1 \preceq \boldsymbol{w}_0$ and $\boldsymbol{\alpha}_1 \neq \boldsymbol{w}_0$, we have that $\boldsymbol{w}_1 = \boldsymbol{w}_0 - \boldsymbol{\alpha}_1 \neq \mathbf{0}$. This contradiction implies that $\theta_2 \leq \theta_1$.

We now examine the solution vectors to the dual problems of $\mathcal{P}.1$, $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$. We know that $\left\|\boldsymbol{\alpha}_1\right\|_1 = \lambda_1$ so that $\left\|\boldsymbol{w}_0 - \boldsymbol{\alpha}_1\right\|_1 > \lambda_2$ and hence $\boldsymbol{\alpha}_2$ is at the boundary $\left\|\boldsymbol{\alpha}_2\right\|_1 = \lambda_2$ (see again Duchi et al. 2008). Furthermore, the sum of the these vectors is

$$\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 = [\boldsymbol{w}_0 - \theta_1]_+ + \left[\boldsymbol{w}_0 - [\boldsymbol{w}_0 - \theta_1]_+ - \theta_2\right]_+ \ . \tag{32}$$

Let v denote a component of $\boldsymbol{w}_0$ greater than θ_1. For any such component the right hand side of Eq. (32) amounts to

$$[v - (v - \theta_1) - \theta_2]_+ + [v - \theta_1]_+ = [\theta_1 - \theta_2]_+ + v - \theta_1 = v - \theta_1 = [v - \theta_1]_+ \ ,$$

where we used the fact that $\theta_2 \leq \theta_1$ to eliminate the term $[\theta_1 - \theta_2]_+$. Next, let u denote a component of $\boldsymbol{w}_0$ smaller than θ_1. In this case, the right hand side of Eq. (32) amounts to $[u - 0 - \theta_2]_+ + 0 = [u - \theta_2]_+$. Recapping, the end result is that the vector sum $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2$ equals $[\boldsymbol{w}_0 - \theta_2]_+$. Moreover, $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ are in $\mathbb{R}^n_+$ as we assumed that $\boldsymbol{w}_0 \succeq \mathbf{0}$, and thus

$$\left\|\,[\boldsymbol{w}_0 - \theta_2]_+\,\right\|_1 = \left\|\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2\right\|_1 = \lambda_1 + \lambda_2 \ . \tag{33}$$

We now show that $\mathcal{P}.2$ has the same dual solution as the sequential updates above. The dual of $\mathcal{P}.2$ is

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \ \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{w}_0\right\|_2^2 \ \text{ s.t. } \ \left\|\boldsymbol{\alpha}\right\|_1 \leq \lambda_1 + \lambda_2.$$

Denoting by $\boldsymbol{\alpha}_0$ the solution of the above dual problem, we have $\boldsymbol{w}^\star = \boldsymbol{w}_0 - \boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_0 = [\boldsymbol{w}_0 - \theta]_+$ for some $\theta \in \mathbb{R}_+$. Examining the norm of $\boldsymbol{\alpha}_0$ we obtain that

$$\left\|\boldsymbol{\alpha}_0\right\|_1 = \left\|[\boldsymbol{w}_0 - \theta]_+\right\|_1 = \lambda_1 + \lambda_2 \tag{34}$$

because we assumed that $\left\|\boldsymbol{w}_0\right\|_1 > \lambda_1 + \lambda_2$. We can view the terms $\left\|[\boldsymbol{w}_0 - \theta_2]_+\right\|_1$ from Eq. (33) and $\left\|[\boldsymbol{w}_0 - \theta]_+\right\|_1$ from Eq. (34) as functions of θ_2 and θ, respectively. The functions are strictly decreasing functions of θ and θ_2 over the interval $[0, \left\|\boldsymbol{w}_0\right\|_\infty]$. Therefore, they are invertible for $0 < \lambda_1 + \lambda_2 < \left\|\boldsymbol{w}_0\right\|_1$. Since $\left\|[\boldsymbol{w}_0 - \theta]_+\right\|_1 = \left\|[\boldsymbol{w}_0 - \theta_2]_+\right\|_1$, we must have $\theta_2 = \theta$. Recall that the solution of Eq. (31) is $\boldsymbol{w}^\star = \min\{\boldsymbol{w}_0, \theta\}$, and the solution of the sequential update induced by Eq. (29) and Eq. (30) is $\min\{\min\{\boldsymbol{w}_0, \theta_1\}, \theta_2\} = \min\{\boldsymbol{w}_0, \theta_2\}$. The programs $\mathcal{P}.1$ and $\mathcal{P}.2$ therefore result in the same vector $\min\{\boldsymbol{w}_0, \theta_2\} = \min\{\boldsymbol{w}_0, \theta\}$ and their induced updates are equivalent.

We now examine the case when $\left\|\boldsymbol{w}_0\right\|_1 \leq \lambda_1 + \lambda_2$. If the 1-norm of $\boldsymbol{w}_0$ is also smaller than λ_1, $\left\|\boldsymbol{w}_0\right\|_1 \leq \lambda_1$, then the dual solution for the first step of $\mathcal{P}.1$ is $\boldsymbol{\alpha}_1 = \boldsymbol{w}_0$, which makes $\boldsymbol{w}_1 = \boldsymbol{w}_0 - \boldsymbol{\alpha}_1 = \mathbf{0}$ and hence $\boldsymbol{w}_2 = \mathbf{0}$. The dual solution for the combined problem is clearly $\boldsymbol{\alpha}_0 = \boldsymbol{w}_0$; again, $\boldsymbol{w}^\star = \boldsymbol{w}_0 - \boldsymbol{\alpha}_0 = \mathbf{0}$. We are thus left with the case $\lambda_1 < \left\|\boldsymbol{w}_0\right\|_1 \leq \lambda_1 + \lambda_2$. We straightforwardly get that the solution to Eq. (31) is $\boldsymbol{w}^\star = \mathbf{0}$. We now prove that the iterated solution obtained by $\mathcal{P}.1$ results in the zero vector as well. First, consider the dual solution $\boldsymbol{\alpha}_1$, which is the minimizer of $\left\|\boldsymbol{\alpha} - \boldsymbol{w}_0\right\|^2$ subject to $\left\|\boldsymbol{\alpha}\right\|_1 \leq \lambda_1$. Since $\boldsymbol{\alpha}_1 = [\boldsymbol{w}_0 - \theta_1]_+$ for

some $\theta_1 \geq 0$, we know that each component of $\boldsymbol{\alpha}_1$ is between zero and its corresponding component in $\boldsymbol{w}_0$, therefore, $\|\boldsymbol{w}_0 - \boldsymbol{\alpha}_1\|_1 = \|\boldsymbol{w}_0\|_1 - \|\boldsymbol{\alpha}\|_1 = \|\boldsymbol{w}_0\|_1 - \lambda_1 \leq \lambda_2$. The dual of the second step of $\mathcal{P}.1$ distills to the minimization $\frac{1}{2}\|\boldsymbol{\alpha} - (\boldsymbol{w}_0 - \boldsymbol{\alpha}_1)\|^2$ subject to $\|\boldsymbol{\alpha}\|_1 \leq \lambda_2$. Since we showed that $\|\boldsymbol{w}_0 - \boldsymbol{\alpha}\|_1 \leq \lambda_2$, we get $\boldsymbol{\alpha}_2 = \boldsymbol{w}_0 - \boldsymbol{\alpha}_1$. This means that $\theta_2 = 0$. Recall that the solution of $\mathcal{P}.1$ is $\min\{\boldsymbol{w}_0, \theta_2\}$, which amounts to the zero vector when $\theta_2 = 0$. We have thus showed that both optimization problems result in the zero vector. This proves the equivalence of $\mathcal{P}.1$ and $\mathcal{P}.2$ for $q = \infty$. ■

The algorithmic consequence of Proposition 11 is that it is possible to perform a lazy update on each iteration by omitting the terms of $\boldsymbol{w}_t$ (or whole rows of the matrix W_t when using mixed-norms) that are outside the support of $\boldsymbol{g}_t^f$, the gradient of the loss at iteration t. However, we do need to maintain the step-sizes used on each iteration and have them readily available on future rounds when we need to update coordinates of $\boldsymbol{w}$ or W that were not updated in previous rounds. Let Λ_t denote the sum of the step sizes times regularization multipliers $\lambda\eta_t$ used from round 1 through t. Then a simple algebraic manipulation yields that instead of solving

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_t\|_2^2 + \lambda\eta_t \|\boldsymbol{w}\|_q \right\}$$

repeatedly when $\boldsymbol{w}_t$ is not being updated, we can simply cache the last time t_0 that $\boldsymbol{w}$ (or a coordinate in $\boldsymbol{w}$ or a row from W) was updated and, when it is needed, solve

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_t\|_2^2 + (\Lambda_t - \Lambda_{t_0})\|\boldsymbol{w}\|_q \right\} \quad .$$

The advantage of the lazy evaluation is pronounced when using mixed-norm regularization as it lets us avoid updating entire rows so long as the row index corresponds to a zero entry of the gradient $\boldsymbol{g}_t^f$. We would like to note that the lazy evaluation due to Proposition 11 includes as a special case the efficient implementation for ℓ_1-regularized updates first outlined by Langford et al. (2008). In sum, at the expense of keeping a time stamp t for each entry of $\boldsymbol{w}$ or row of W and maintaining a list of the cumulative sums $\Lambda_1, \Lambda_2, \ldots$, we can get $O(s)$ updates of $\boldsymbol{w}$ when the gradient $\boldsymbol{g}_t^f$ is s-sparse, that is, it has only s non-zero components.

7. Experiments

In this section we describe the results of experiments we performed whose goal are to demonstrate the merits and underscore a few weaknesses of FOBOS. To that end, we also evaluate specific instantiations of FOBOS with respect to several state-of-the-art optimizers and projected subgradient methods on different learning problems. In the experiments that focus on efficiency and speed of convergence, we evaluate the methods in terms of their number of operations, which is approximately the number of floating point operations each method performs. We believe that this metric offers a fair comparison of the different algorithms as it lifts the need to cope with specific code optimization such as cache locality or different costs per iteration of each of the methods.

7.1 Sensitivity Experiments

We begin our experiments by performing a sensitivity analysis of FOBOS. We perform some of the analysis in later sections during our comparisons to other methods, but we discuss the bulk of it here. We focus on two tasks in our sensitivity experiments: minimizing the hinge loss (used in Support Vector Machines) with ℓ_2^2 regularization and minimizing the ℓ_1-regularized logistic loss. These set the loss function f as

$$f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} [1 - y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle]_+ + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \quad \text{and} \quad f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + e^{-y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle}\right) + \lambda\|\boldsymbol{w}\|_1$$

respectively. Note that both loss functions have subgradient sets bounded by $\frac{1}{n}\sum_{i=1}^{n}\|y_i \boldsymbol{x}_i\|_2$. Therefore, if all the instances are of bounded norm, so are the subgradients of the empirical loss functions.

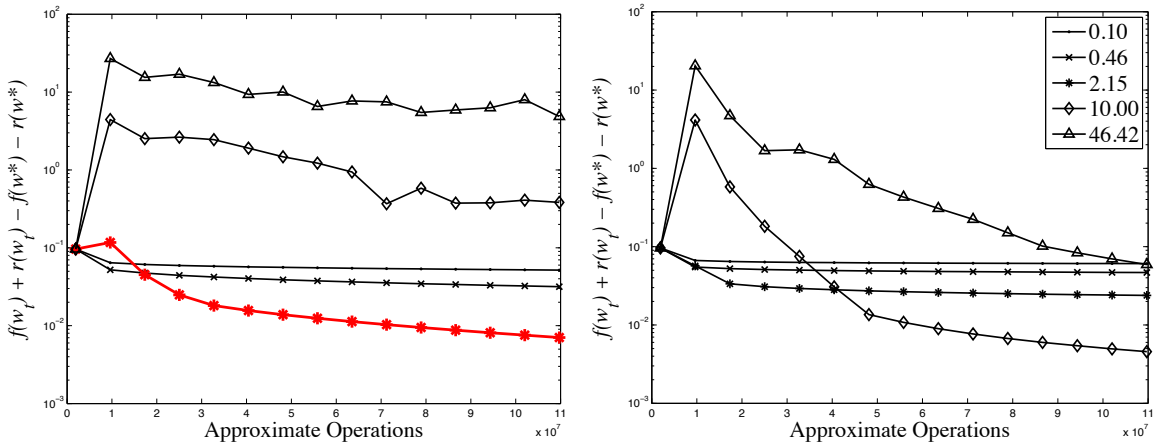


Figure 1: Sensitivity of deterministic FOBOS to initial step size on logistic regression. Key is initial step size. Left: $\eta_t \propto 1/\sqrt{t}$. Right: $\eta_t \propto 1/t$.

We perform analysis using dimensions $d \in \{50, 100, 200, 400, 800\}$ and data set sizes $n \in \{200, 400\}$. We investigate the effect of correlation of the features x_{ij} with one another by generating uncorrelated, moderately correlated, and highly correlated features. Specifically, to generate feature vectors $\boldsymbol{x}_i \in \mathbb{R}^d$, we sample n random vectors $\boldsymbol{z}_i \sim N(0, I)$, $\boldsymbol{z}_i \in \mathbb{R}^d$. We then construct a random covariance matrix Σ whose correlations $|\rho_{ij}| = |\Sigma_{ij}|/\sqrt{\Sigma_{ii}\Sigma_{jj}}$ have mean .2 for the moderately correlated experiments and .3 for the highly correlated experiments (on the highly correlated data, more than one-tenth of the features were more than 80% correlated). To get $\boldsymbol{x}$, we then set $\boldsymbol{x}_i = L\boldsymbol{z}_i$, where L is the Cholesky decomposition of Σ (the identity in the uncorrelated case), and normalize $\boldsymbol{x}_i$ to have $\|\boldsymbol{x}_i\|_\infty = 1$. We compared different stochastic gradient estimators that were based on varying sample sizes: a single example, 10% of the training set, 20% of the training set, and deterministic gradient using the entire data set. We also tested ten different initial step sizes. However, we give in the graphs results for only a subset of the initial steps that reveals the overall dependency on the step size. Further, we also checked three schemes for decaying the step size: $\eta_t \propto 1/t$, $\eta_t \propto 1/\sqrt{t}$, and $\eta_t \propto 1$ (constant step size). We discuss the results attained by constant step sizes only qualitatively, though, to keep the clarity of the figures. When η_t was a constant we divided the initial step size by $\sqrt{T}$, the total number of iterations being taken. We performed each experiment 10 times and averaged the results.

We distill the large number of experiments into a few figures here, deferring some of the analysis to the sequel. Thus in this section we focus on the case when $n = 400$ and $d = 800$, since the experiments with other training set sizes and dimensions gave qualitatively similar results. Most of the results in this section focus on the consequences of the initial step size η_1, though we also discuss different schedules of the learning rate and the sample size for computing the subgradients. In each experiment, we set $\lambda = .25/\sqrt{n}$, which in the logistic case gave roughly 50% sparsity. Before our discussion, we note that we can bound the ℓ_2-norm of $\boldsymbol{w}^\star$ for both the logistic and the hinge loss. In the case of the logistic, we have $\lambda\|\boldsymbol{w}^\star\|_2 \le \lambda\|\boldsymbol{w}^\star\|_1 \le f(\boldsymbol{0}) = \log 2$. Similarly, for the hinge loss we have $\frac{\lambda}{2}\|\boldsymbol{w}^\star\|_2^2 \le f(\boldsymbol{0}) = 1$. In both cases we can bound G by the norm of the $\|\boldsymbol{x}_i\|$, which is in our settings approximately 9. Thus, looking at the bounds from Sec. 3 and Sec. 4, when $\eta_t \propto 1/\sqrt{T}$, the initial step size amounts to $\eta_1 \approx D/\sqrt{7}G \approx 2.3$ and when $\eta_t \propto 1/\sqrt{t}$, the initial step should be $\eta_1 \approx D/4G \approx 1.5$ for logistic regression. For the hinge loss, when $\eta_t \propto 1/\sqrt{t}$, and the initial step ends being $\eta_1 \approx D/4G \approx .35$. We see in the sequel that these approximate step sizes yield results competitive with the best initial step sizes, which can be found only in hindsight.

We begin by considering the effect of initial step size for ℓ_1-regularized logistic regression. We plot results for the moderately correlated data sets, as we investigate the effect of correlation later on. The results are given in Figures 1 and 2, where we plot the objective value at each time step minus the optimal objective

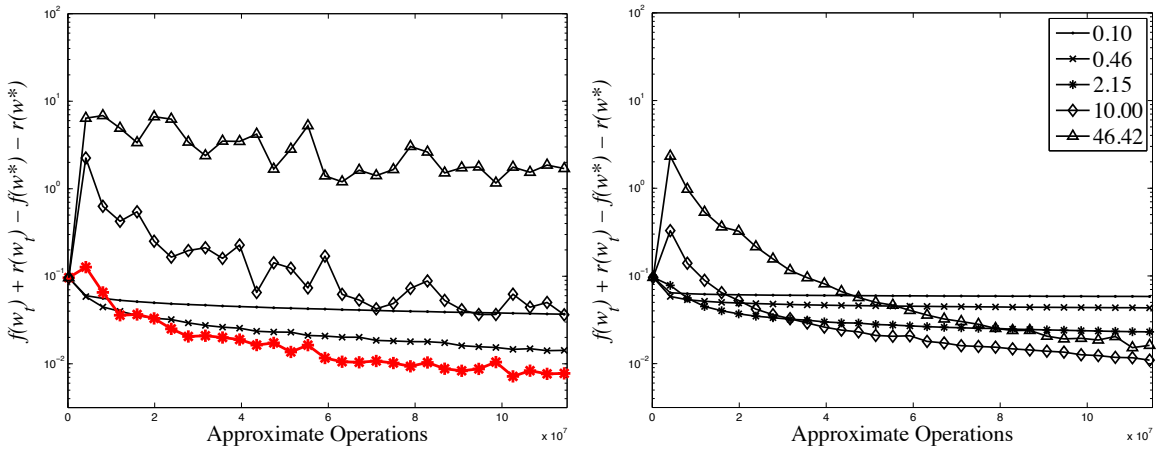


Figure 2: Sensitivity of stochastic FOBOS to initial step size on logistic regression. Key is initial step size. Left: $\eta_t \propto 1/\sqrt{t}$. Right: $\eta_t \propto 1/t$.

value attained, $f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t) - f(\boldsymbol{w}^\star) - r(\boldsymbol{w}^\star)$. We plot the function values of the initial step size that was chosen automatically, by estimating D and G as described above, in bold red in the figures. Interestingly, the left side of Fig. 1 suggests that the best performing initial steps when $\eta_t \propto 1/\sqrt{t}$ are near 1.5: $\eta_1 = 2.15$ gives good regret (we also saw that $\eta_1 = 4.6$ performed well, but do not include it in the plot). In Fig. 2, we see similar behavior for stochastic FOBOS when using 10% of the samples to estimate the gradient. Though we do not plot these results, the stochastic case with constant step sizes also performs well when the step sizes are $\eta_t \approx 2.2/\sqrt{T}$. The deterministic variant of FOBOS could not not take as many steps in the allotted time as the stochastic versions, so its performance was not competitive. All methods are somewhat sensitive to initial step size. Nonetheless, for these types of learning problems it seems plausible to estimate an initial step size based on the arguments given in the previous paragraph, especially when using the step rate of $\eta_t \propto 1/\sqrt{t}$ that was suggested by the analysis in Sec. 4.

We performed similar experiments with the hinge loss with ℓ_2^2 regularization. As the objective is strongly convex,[2] our analysis from Sec. 4 and Theorem 8 suggests a step rate of $\eta_t = 1/\lambda t$. From the right plot in Fig. 3, we see that the best performing step sizes where the two largest, which is consistent with our analysis since $1/\lambda = 80$. For the the $1/\sqrt{t}$ steps, which we present on the left of Fig. 3 respectively, such initial step sizes are too large. However, we can see again that the approximation suggested by our arguments above gives good performance. In sum, it seems that while FOBOS and stochastic FOBOS are fairly sensitive to initial step sizes and rate schedule, our theoretical results from the previous sections give relatively good initial step size heuristics. We will see similar behavior in the sequel.

7.2 Comparison to Subgradient Optimizers

We now move on to the description of experiments using FOBOS to solve ℓ_2^2-regularized learning problems, focusing on comparison to the state-of-the-art subgradient optimizer Pegasos (Shalev-Shwartz et al., 2007). Pegasos was originally implemented and evaluated on Support Vector Machine (SVM) problems by using the hinge-loss as the empirical loss function along with an ℓ_2^2 regularization term. Nonetheless, Pegasos can be rather simply extended to the binary logistic loss function. We thus experimented with both the hinge and logistic loss functions. To generate data for our experiments, we chose a vector $\boldsymbol{w}$ with entries distributed normally with a zero mean and unit variance, while randomly zeroing 50% of the entries in the vector. The

2. We assume there is no bias term in the objective, since any optimization method must deal with this so we find it outside the scope of the paper.

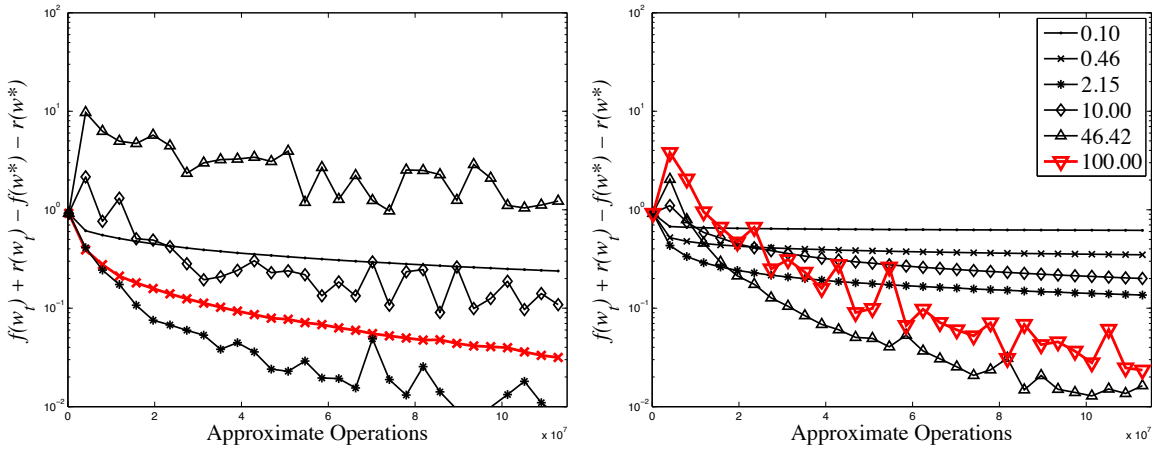


Figure 3: Sensitivity of stochastic methods to initial step size on hinge loss minimization with ℓ_2^2-regularization. Key is initial step size. Left: $\eta_t \propto 1/\sqrt{t}$. Right: $\eta_t \propto 1/t$.

examples $\boldsymbol{x}_i \in \mathbb{R}^d$ were also chosen at random with entries i.i.d. normally distributed. We also performed experiments using correlated data. The results attained on correlated data were similar, so we do not report them in our comparison to Pegasos. To generate target values, we set $y_i = \text{sign}(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle)$, and negated the sign of 10% of the examples to add label noise. In all experiments, we used $n = 1000$ training examples of dimension $d = 400$.

The graphs of Fig. 4 show (on a log-scale) the objective function, namely, the regularized empirical loss of the algorithms, minus the optimal value of the objective function. These results were averaged over 20 independent runs of the algorithms. In all experiments with a regularization of the form $\frac{1}{2}\lambda\|\boldsymbol{w}\|_2^2$, we used step sizes of the form $\eta_t = 1/(\lambda t)$ to achieve the logarithmic regret bound of Sec. 4. The left graph of Fig. 4 conveys that FOBOS performs comparably to Pegasos on hinge (SVM) loss. Both algorithms quickly approach the optimal value. In this experiment we let both Pegasos and FOBOS employ a projection after each gradient step onto a ℓ_2 norm ball in which $\boldsymbol{w}^\star$ must lie (see Shalev-Shwartz et al. 2007 and the discussion following the proof of Theorem 2). However, in the experiment corresponding to the right plot of Fig. 4, we eliminated the additional projection step and ran the algorithms with the logistic loss. In this case, FOBOS slightly outperforms Pegasos. We hypothesize that the slightly faster rate of FOBOS is due to the explicit shrinkage that FOBOS performs in the ℓ_2^2 update (see Eq. (20) and Lemma 9).

7.3 Comparison to Other Methods for Smooth Problems

As mentioned in our discussion of related work, many methods have been proposed in the optimization and machine learning literature for minimizing Eq. (1) when f is smooth, particularly when f has a Lipschitz-continuous gradient. For the case of ℓ_1-regularized logistic regression, Koh et al. (2007) propose an efficient interior point method. Tseng and Yun (2007) give analysis of a (block) coordinate descent method that uses approximations to the Hessian matrix and an Armijo-type backtracking line search to solve non-smooth regularized problems with smooth objectives; their method was noted to be effective for ℓ_1/ℓ_2-regularized logistic regression, for example, by Meier et al. (2008). We also compare FOBOS and its stochastic variants to the SPARSA method of Wright et al. (2009), which shares the same update as FOBOS but uses a simple line search strategy to choose the its steps. Note that none of these methods apply when f is non-smooth. Lastly, we compare FOBOS to projected-gradient methods. For ℓ_1-regularized problems, Duchi et al. (2008) show how to compute projections to an ℓ_1-ball in linear time, and Schmidt et al. (2009) extend the method to show that projection of a matrix $W \in \mathbb{R}^{d \times k}$ to an ℓ_1/ℓ_2-constraint can be computed in $O(dk)$ time. To compare

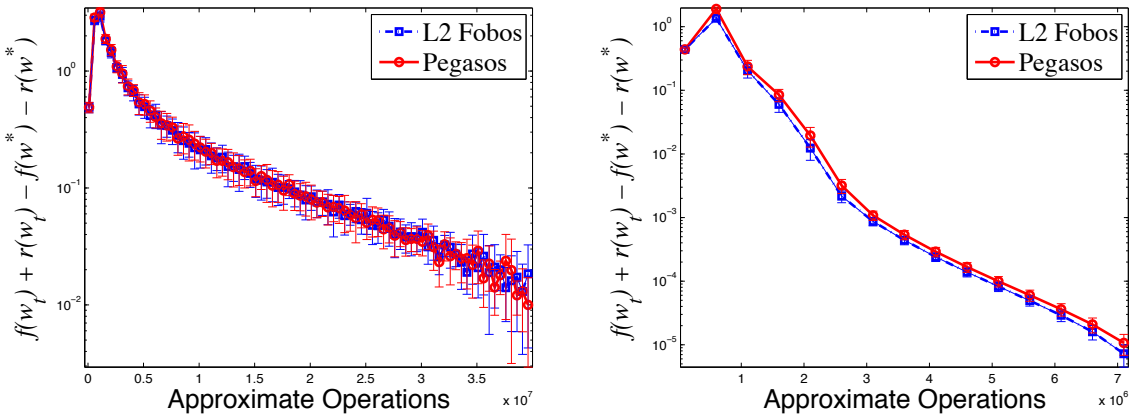


Figure 4: Comparison of FOBOS with Pegasos on the SVM (left) and logistic regression (right). The right hand side plot shows the performance of the algorithms without a projection step.

our methods to projected gradient methods, we first solve the regularized version of the problem. We then constrain the norm of $\boldsymbol{w}$ or the mixed-norm of W to lie within a ball of radius $\lambda\|\boldsymbol{w}^\star\|_1$ or $\lambda\|W^\star\|_{\ell_1/\ell_2}$.

We compared FOBOS to each of the above methods on many different synthetic data sets, and we report a few representative results. In our experiments, SPARSA seemed to outperform FOBOS and the projected gradient methods when using full (deterministic) gradient information. The additional function evaluations incurred by the line search in SPARSA seem to be insignificant to runtime, which is a plausible explanation for SPARSA's superior performance. We therefore do not report results for the deterministic versions of FOBOS and projected gradient methods to avoid clutter in the figures.

In the first set of experiments, we compare FOBOS's performance on ℓ_1-regularized logistic regression to each of the above methods. That is, we set $f(\boldsymbol{w})$ to be the average logistic loss and $r(\boldsymbol{w}) = \lambda\|\boldsymbol{w}\|_1$ and use a data set with $n = 1000$ examples and $d = 400$ dimensions. We compare the performance of stochastic FOBOS to the other algorithms in terms of two aspects. The first is the value of λ, which we set to five logarithmically spaced values that gave solution vectors $\boldsymbol{w}^\star$ ranging from 100% non-zero entries to only 5% non-zero entries. The second aspect is on the correlation of the features. We generated random data sets with uncorrelated features, features that were on average 20% correlated with one another, and features that were on average 30% correlated with one another. In the latter case about 350 pairs of features had above 80% correlation (see the description of feature generation at the beginning of the section). We normalized each example $\boldsymbol{x}$ to have features in the range $[-1,1]$. We assigned labels for each example by randomly choosing a 50% sparse vector $\boldsymbol{w}$, setting $y_i = \text{sign}(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)$, and negating 5% of the y values.

The results comparing FOBOS to the other algorithms for different settings of the regularizer λ are in Fig. 5. The y-axis is $f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t) - f(\boldsymbol{w}^\star) - r(\boldsymbol{w}^\star)$, the distance of the current value from the optimal value, and the x-axis is the approximate number of operations (FLOPs) for each method. We used the approximation we derived based on Corollary 7 in our earlier discussion of sensitivity to set the initial step size and used $\eta_t \propto 1/\sqrt{t}$. Tseng and Yun's method requires setting of constants for the backtracking-based line search. We thus use the settings in Meier et al. (2008). In attempt to make the comparisons as fair as possible, we used some of Tseng and Yun's code yet reimplemented the method to take advantage of the specific structure of logistic regression. Similarly, we used the line-search parameters in Wright et al.'s publicly available Matlab code for SPARSA, though we slightly modified their code to handle arbitrary loss functions. From the figure, we see that as λ grows, yielding sparser solutions for $\boldsymbol{w}^\star$, the performance of coordinate descent and especially the interior point method start to degrade relative to the stochastic methods and SPARSA.

In our experiments we found that the stochastic methods were quite resilient to overly-large initial step-sizes, as they quickly took a large number of steps. SPARSA employs an easy to implement and efficient line search, and in general yielded good performance. The coordinate descent method, with its somewhat

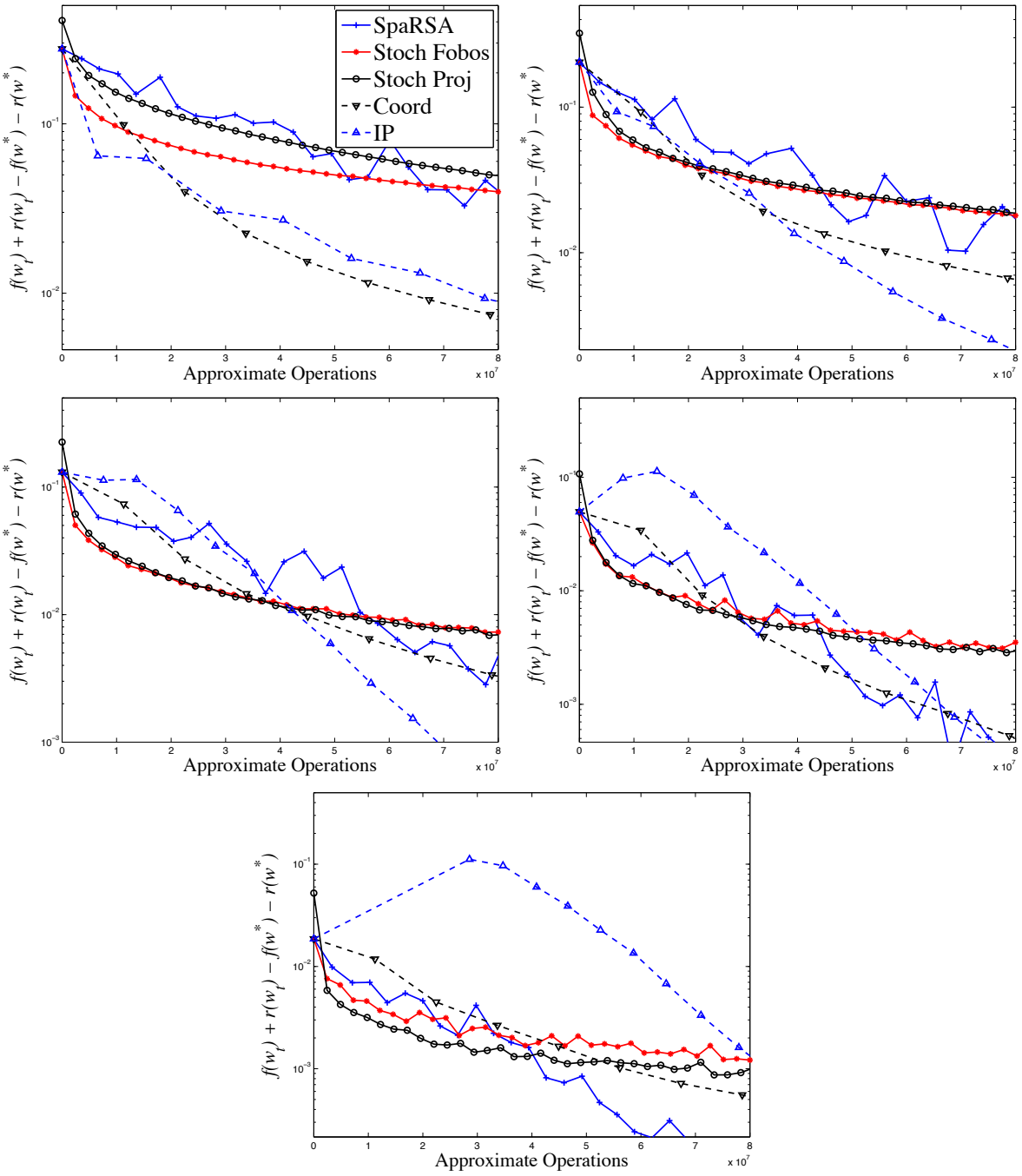


Figure 5: Performance of ℓ_1-regularized logistic regression methods with different settings of λ on correlated synthetic data. Left to right, top to bottom: $\boldsymbol{w}^\star$ has 0% sparsity, $\boldsymbol{w}^\star$ has 25% sparsity, $\boldsymbol{w}^\star$ has 40% sparsity, $\boldsymbol{w}^\star$ has 70% sparsity, and $\boldsymbol{w}^\star$ has 95% sparsity.

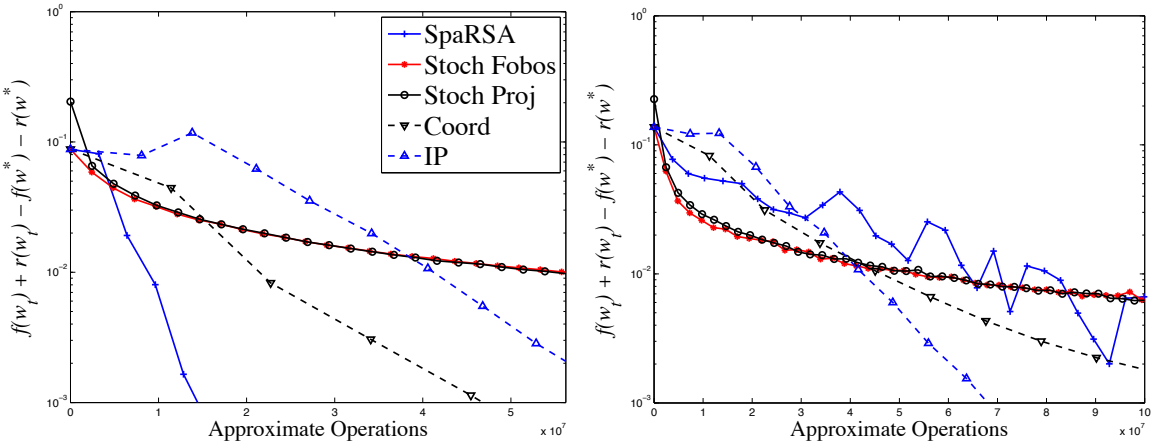


Figure 6: Performance of ℓ_1-regularized logistic regression methods with different correlations on synthetic data. Left: uncorrelated data. Right: highly correlated data.

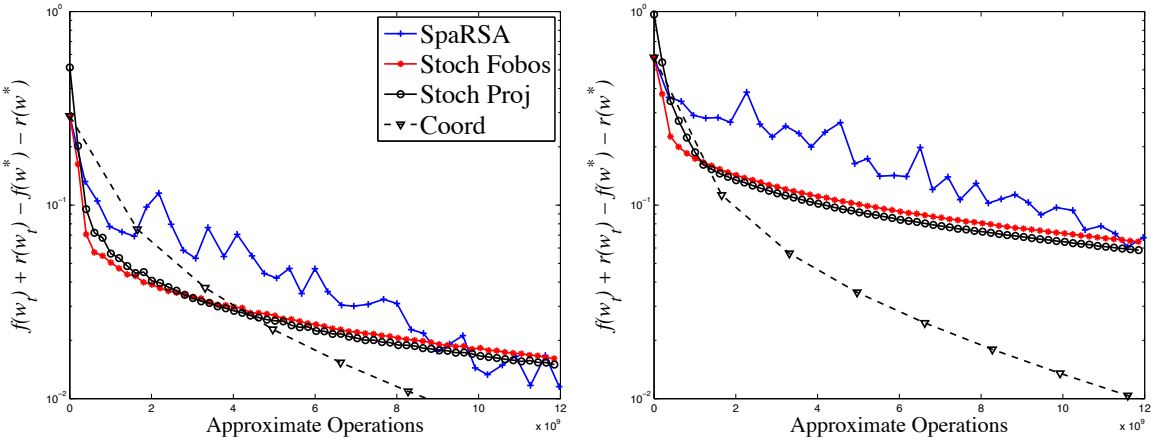


Figure 7: Performance of ℓ_1/ℓ_2-regularized multiclass logistic regression methods with different settings of λ on correlated synthetic data. Left: $\boldsymbol{w}^\star$ has 60% sparsity. Right: $\boldsymbol{w}^\star$ has 30% sparsity.

complicated backtracking line search, was difficult to implement correctly. Therefore, our experiments and experience suggest that SPARSA is likely to be preferred for smooth problems. Nonetheless, stochastic FOBOS quickly obtains a solution within about 10^{-2} of the optimal value. Since the regularized empirical loss serves as a proxy for attaining good generalization performance, we found that in numerous cases this accuracy sufficed to achieve competitive *test* loss.

In Fig. 6 we compare FOBOS to the other methods on data with uncorrelated, moderately correlated, and very correlated features. These plots all have λ set so that $\boldsymbol{w}^\star$ has approximately 40% sparsity. From the plots, we see that stochastic FOBOS and projected gradient actually perform very well on the more correlated data, very quickly getting to within 10^{-2} of the optimal value, though after this they essentially jam. As in the earlier experiments, SPARSA seems to perform quite well for these moderately sized experiments, though the interior point method's performance improves as the features become more correlated.

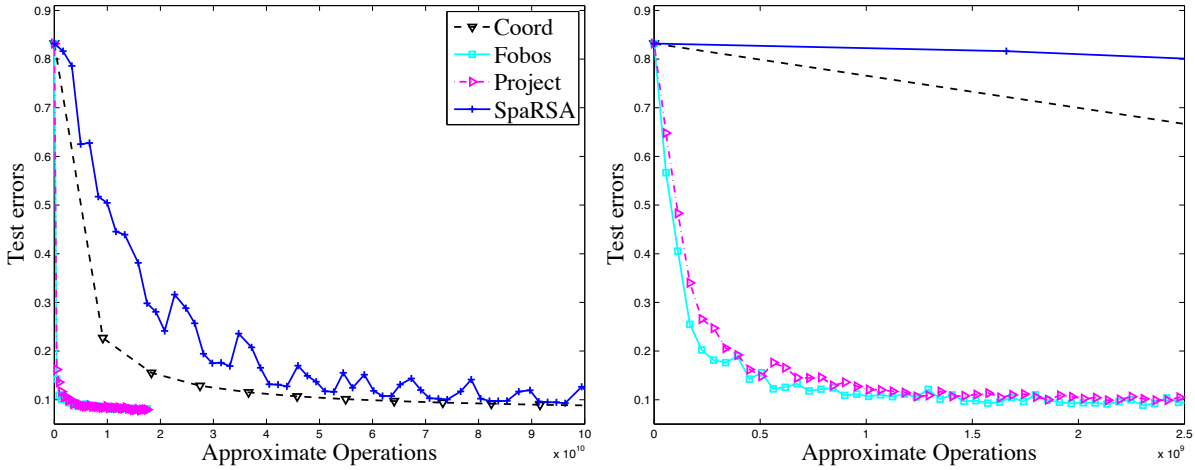


Figure 8: Comparison test error rate of FOBOS, SPARSA, projected gradient, and coordinate descent on MNIST digit recognition data set. Right: magnified view of left plot.

The last set of experiments with synthetic data sets was on mixed-norm-regularized multiclass logistic regression. The objective that we used in this case is

$$\frac{1}{n}\sum_{i=1}^{n}\log\left(1+\sum_{j\neq y_i}e^{\langle x_i, w^j - w^{y_i}\rangle}\right)+\lambda\|W\|_{\ell_1/\ell_q} \quad . \tag{35}$$

In the above equation q represents the norm over rows of the matrix W, and in our experiments it is either 1, 2, or ∞ (in this section, $q=2$). The goal is to classify correctly examples whose labels are in $\{1,\ldots,k\}$ while jointly regularizing entries of the vectors w^j. We used $n=5000$ datapoints of dimension $d=1000$ with $k=10$ classes, meaning we minimize a loss over a matrix $W\in\mathbb{R}^{d\times k}$ with 10000 parameters. To generate data, we sample examples x_i from a normal distribution with moderate correlation, randomly choose a matrix W, and set $y_i=\operatorname{argmax}_j\langle x_i, w^j\rangle$ with 5% label noise. We show results in Fig. 7. In the three figures, vary λ to give solutions $W^\star$ with roughly 60% zero rows, 30% zero rows, and completely non-sparse $W^\star$. From the figures, it is apparent that the stochastic methods, both FOBOS and projected gradient, exhibit very good initial performance but eventually lose to the coordinate descent method in terms of optimization speed. As before, if one is willing to use full gradient information, SPARSA seems a better choice than the deterministic counterpart of FOBOS and projected gradient algorithms. We thus again do not present results for deterministic FOBOS without any line search.

7.4 Experiments with Real Data Sets

Though in the prequel we focus on optimization speed, the main goal in batch learning problems is attaining good test-set error rates rather than rapid minimization of $f(w)+r(w)$. In order to better understand the merits of different optimization methods, we compared the performance of different optimizers on achieving a good test-set error rate on different data sets. Nonetheless, for these tests the contours for the training objective value were qualitatively very similar to the test-set error rate curves. We used the StatLog LandSat Satellite data set (Spiegelhalter and Taylor, 1994), the MNIST handwritten digit database, and a sentiment classification data set (Blitzer et al., 2007).

The MNIST database consists of 60,000 training examples and a 10,000 example test set with 10 classes. We show average results over ten experiments using random 15,000 example subsamples of the training set. For MNIST, each digit is a 28×28 gray scale image z which is represented as a $28^2=784$ dimensional

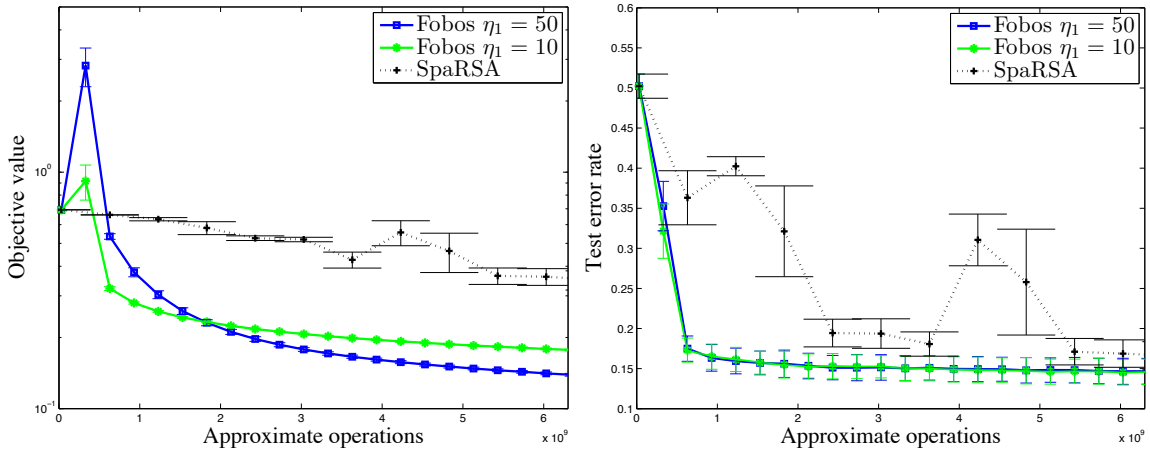


Figure 9: Comparison of FOBOS and SPARSA on sentiment classification task.

vector. Direct linear classifiers do not perform well on this data set. Thus, rather than learning weights for the original features, we learn the weights for a kernel machine with Gaussian kernels, where the value of the j^{th} feature for the i^{th} example is

$$x_{ij} = K(\boldsymbol{z}_i, \boldsymbol{z}_j) \triangleq e^{-\frac{1}{2}\|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2} \quad .$$

We used ℓ_1/ℓ_2 regularization and compared FOBOS, SPARSA, coordinate descent, and projected gradient methods on this test (as well as stochastic gradient versions of FOBOS and projected gradient). The results for deterministic FOBOS and projected gradient were similar to SPARSA, so we do not present them. We also experimented with stochastic group sizes of 100 and 200 examples for FOBOS, but the results were similar, so we plot only the results from the 100 example runs. As before, we used the ℓ_1/ℓ_2 norm of the solution vectors for FOBOS, SPARSA, and coordinate descent as the constrained value for the projected gradient method. For each of the gradient methods, we estimated the diameter D and maximum gradient G as in the synthetic experiments, which led us to use a step size of $\eta_t = 30/\sqrt{t}$. The test set error rate as a function of number of operations for each of the methods are shown in Fig. 8. From Fig. 8, it is clear that the stochastic gradient methods (FOBOS and projected gradient) were significantly faster than any of the other methods. Before the coordinate descent method has even visited every coordinate once, stochastic FOBOS (and similarly stochastic projected gradient) have attained the minimal test-set error. The inferior performance of coordinate descent and deterministic gradient methods can be largely attributed to the need to exhaustively scan the data set. Even if we use only a subset of 15,000 examples, it takes a nontrivial amount of time to simply handle each example. Moreover, the objective values attained during training are qualitatively very similar to the test loss, so that stochastic FOBOS *much* more quickly reduces the training objective than the deterministic methods.

We also performed experiments on a data set that was very qualitatively different from the MNIST and LandSAT data sets. For this experiment, we used the multi-domain sentiment data set of Blitzer et al. (2007), which consists of product reviews taken from Amazon.com for many product types. The prediction task is to decide whether an article is a positive or negative review. The features are bigrams that take values in $\{0,1\}$, totalling about 630,000 binary features. In any particular example, at most a few thousand features are non-zero. We used 10,000 examples in each experiment and performed 10 repetitions while holding out 1000 of the examples as a test set and using 9000 of the examples for training. We used ℓ_1-regularized logistic regression and set $\lambda = 3 \cdot 10^{-5}$, which gave the best generalization performance and resulted in roughly 5% non-zeros in the final vector $\boldsymbol{w}$. We compare stochastic FOBOS to SPARSA in Fig. 9, since the projected gradient method is much slower than FOBOS (detailed in the sequel). For FOBOS we use 900 examples to compute each stochastic gradient. We use two different initial step sizes, one estimated using the approximation described earlier and a second where we scale it by $1/5$. The left plot in Fig. 9 shows the

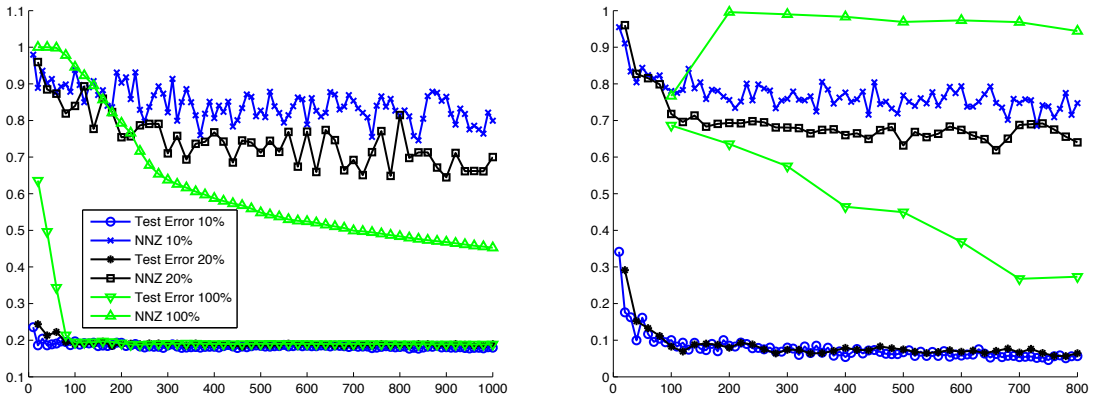


Figure 10: Left: FOBOS sparsity and test error for LandSat data set with ℓ_1-regularization. Right: FOBOS sparsity and test error for MNIST data set with ℓ_1/ℓ_2-regularization. Key is identical for both plots.

training objective value as a function of the number of operations for each of the three methods as well as error bars equal to the standard deviation of the objective. The right plot shows the error rates on the test sets. The behavior in the experiment is similar to that in Fig. 8, where the stochastic methods very quickly attain a small test error. Effectively, before SPARSA finishes two steps, the stochastic methods have arrived at approximate solutions that attain the minimal test set error rates.

We now change our focus from training time to the attained sparsity levels for multiclass classification with ℓ_1, ℓ_1/ℓ_2, and ℓ_1/ℓ_∞ regularization on MNIST and the StatLog LandSat data set. For the LandSat data set we attempt to classify 3×3 neighborhoods of pixels in a satellite image as a particular type of ground, and we expanded the input 36 features into 1296 features by taking the product of all features.

In the left plot of Fig. 10, we show the test set error and sparsity level of W as a function of training time (100 times the number of single-example gradient calculations) for the ℓ_1-regularized multiclass logistic loss with 720 training examples. The green lines show results for using all 720 examples to calculate the gradient, black using 20% of the examples, and blue using 10% of the examples to perform stochastic gradient. Each used the same learning rate $\eta_t \propto 1/\sqrt{t}$, and the reported results are averaged over 5 independent runs with different training data sets. The righthand figure shows a similar plot for training FOBOS on the MNIST data set with ℓ_1/ℓ_2-regularization. The objective value in training has a similar contour to the test loss. As expected, FOBOS with stochastic gradient descent gets to its minimum test classification error, and as the training set size increases this behavior is consistent. However, the deterministic version increases the level of sparsity throughout its run, while the stochastic-gradient version has highly variable sparsity levels and does not give solutions as sparse as the deterministic counterpart. We saw similar behavior when using stochastic versus deterministic projected gradient methods. The slowness of the deterministic gradient means that we do not see the sparsification immediately in larger tests; nonetheless, for longer training times similar sparsifying behavior emerges.

As yet, we do not have a compelling justification for the difference in sparsity levels between stochastic and deterministic gradient FOBOS. We give here some intuitive arguments, leaving a more formal analysis to future work. We develop one possible explanation by exploring the effect of taking a stochastic gradient step starting from the true solution vector $\boldsymbol{w}^\star$. Consider ℓ_1-regularized FOBOS with regularization multiplier λ. Let $\boldsymbol{g}^f$ be the gradient of $f(\boldsymbol{w}^\star)$. For j such that $w_j^\star > 0$, we have $g_j^f = -\lambda$, for j with $w_j^\star < 0$, $g_j^f = \lambda$, and for zero entries in $\boldsymbol{w}^\star$, we have $g_j^f \in [-\lambda, \lambda]$. The FOBOS step then amounts to

$w_j^+ = \text{sign}(w_j^\star)\left[|w_j^\star - \eta_t g_j^f| - \eta_t\lambda\right]_+$, which by inspection simply yields $\boldsymbol{w}^\star$. Now suppose that instead of $\boldsymbol{g}^f$, we use a stochastic estimate $\tilde{\boldsymbol{g}}^f$ of $\boldsymbol{g}^f$. Then the probability that the update w_j^+ of $w_j^\star$ is zero is

$$\mathbb{P}\left(\left|w_j^\star - \eta_t \tilde{g}_j^f\right| \leq \eta_t\lambda\right) = \mathbb{P}\left(\tilde{g}_j^f \in \left[\frac{w_j^\star}{\eta_t} - \lambda, \frac{w_j^\star}{\eta_t} + \lambda\right]\right) \quad .$$

When $w_j^\star = 0$, the probability is simply $\mathbb{P}(\tilde{g}_j^f \in [-\lambda, \lambda])$, which does not change as a function of η_t. However, when $w_j^\star > 0$, we have $E[\tilde{g}_j^f] = -\lambda$, while $w_j^\star/\eta_t \to \infty$ as η_t shrinks (and analogously for $w_j^\star < 0$). In essence, the probability of a non-zero parameter staying non-zero is high, however, the probability of a zero parameter staying zero is constant. Intuitively, then, we expect that stochastic gradient descent will result in more non-zero coefficients than the deterministic variant of FOBOS.

7.5 Experiments with Sparse Gradient Vectors

In this section, we consider the implications of Proposition 11 for learning with sparse data. We show that it is much more efficient to use the updates described in Proposition 11 than to maintain ℓ_1-constraints on $\boldsymbol{w}$ as in Duchi et al. (2008). Intuitively, the former requires rather simple bookkeeping for maintaining the sum Λ_t discussed in Sec. 6, and it is significantly easier to implement and more efficient than Duchi et al.'s red-black tree-based implementation. Indeed, whereas a red-black tree requires at least a thousand of lines of code for its balancing, joining, and splitting operations, the efficient FOBOS updates require fewer than 100 lines of code.

We simulated updates to a weight vector $\boldsymbol{w}$ with a sparse gradient $\boldsymbol{g}_t^f$ for different dimensions d of $\boldsymbol{w} \in \mathbb{R}^d$ and different sparsity levels s for $\boldsymbol{g}_t^f$ with $\text{card}(\boldsymbol{g}_t^f) = s$. To do so, we generate a completely dense $\boldsymbol{w}$ whose ℓ_1-norm is at most a pre-specified value b and add a random vector $\boldsymbol{g}$ to $\boldsymbol{w}$ with s non-zeros. We then either project $\boldsymbol{w} + \boldsymbol{g}$ back to the constraint $\|\boldsymbol{w}\|_1 \leq b$ using the algorithm of Duchi et al. (2008) or perform a FOBOS update to $\boldsymbol{w} + \boldsymbol{g}$ using the algorithm in Sec. 6. We chose λ in the FOBOS update to give approximately the same sparsity as the constraint on b. In Table 1 we report timing results averaged over 100 independent experiments for different dimensions d of $\boldsymbol{w}$ and different cardinalities s of $\boldsymbol{g}$. Though theoretically the sparse FOBOS updates should have no dependence on the dimension d, we found that cache locality can play a factor when performing updates to larger dimensional vectors. Nonetheless, it is clear from the table that the efficient FOBOS step is on the order of ten to twenty times faster than its projected counterpart. Furthermore, the sparse FOBOS updates apply equally as well to mixed norm regularization, and while there are efficient algorithms for both projection to both ℓ_1/ℓ_2 and ℓ_1/ℓ_∞ balls (Schmidt et al., 2009; Quattoni et al., 2009), they are more complicated than the FOBOS steps. Lastly, though it may be possible to extend the efficient data structures of Duchi et al. (2008) to the ℓ_1/ℓ_2 case, there is no known algorithm for efficient projections with sparse updates to an ℓ_1/ℓ_∞ constraint.

Dimension d	$s = 5000$		$s = 10000$		$s = 20000$	
	Project	FOBOS	Project	FOBOS	Project	FOBOS
$5 \cdot 10^4$	0.72	0.07	2.12	0.12	4.53	0.23
$2 \cdot 10^5$	0.80	0.10	2.06	0.16	5.09	0.34
$8 \cdot 10^5$	0.86	0.15	2.22	0.17	5.34	0.39
$3.2 \cdot 10^6$	1.07	0.13	2.75	0.16	6.31	0.52
$6.4 \cdot 10^6$	1.20	0.10	2.83	0.29	6.62	0.48

Table 1: Comparison of the average time (in hundredths of a second) required to compute projection of $\boldsymbol{w} + \boldsymbol{g}$ onto an ℓ_1-constraint to the analogous update required by the FOBOS step.

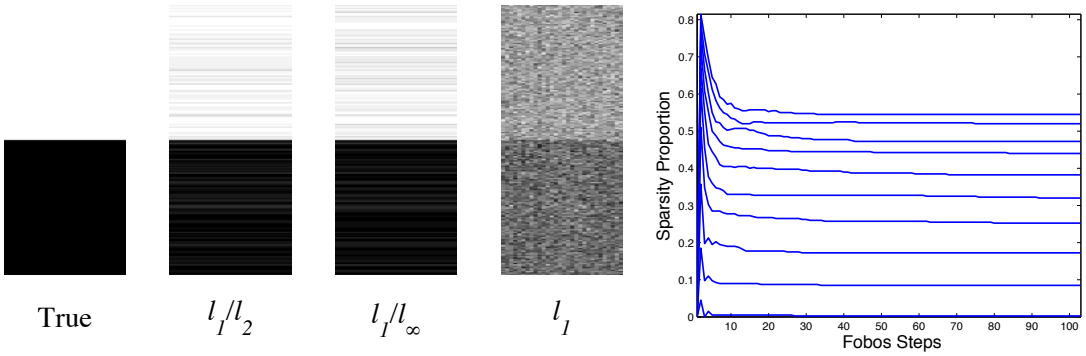


Figure 11: The sparsity patterns attained by FOBOS using mixed-norm and ℓ_1 regularization for a multiclass logistic regression problem.

7.6 Effects of Regularization

Our final experiments focus mostly on sparsity recovery of the different regularizers and their effects on test-set performance. While these are somewhat orthogonal to the previous experiments, we believe there is a relative paucity of investigation of the effects of mixed-norm regularization on classifier performance.

As a verification experiment of FOBOS with a mixed-norm regularizer, we solved a multiclass logistic regression problem whose objective is given in Eq. (35). To solve this task, we randomly generated a matrix W of dimension 200×30. The instances had $d = 200$ dimensions, the number of classes was $k = 30$, and we zeroed out the first 100 rows of W. We next generated $n = 1000$ samples $\boldsymbol{x}_i \in \mathbb{R}^d$ with zero mean and unit variance. We set $y_i = \operatorname{argmax}_j \left\langle \boldsymbol{x}_i, \boldsymbol{w}^j \right\rangle$ and added 10% label noise. We then used FOBOS to find an approximate minimizer of the objective defined by Eq. (35).

To compare the effects of different regularizers, we minimized Eq. (35) using ℓ_1/ℓ_1, ℓ_1/ℓ_2, and ℓ_1/ℓ_∞ regularization. We repeated the experiment 20 times with different randomly selected W that had the same sparsity pattern. On the left side of Fig. 11 we illustrate the sparsity pattern (far left) of the weight vector that generated the data and color-coded maps of the sparsity patterns learned using FOBOS with the different regularization schemes. The colors indicate the fraction of times a weight of W was set to be zero. A white color indicates that the weight was found (or selected) to be zero in all of the experiments while a black color means that it was never zero. The regularization value λ was set so that the learned matrix W would have approximately 50% zero weights. From Fig. 11, we see that both ℓ_1/ℓ_2 and ℓ_1/ℓ_∞ were capable of zeroing entire rows of parameters and often learned a sparsity pattern that was close to the sparsity pattern of the matrix that was used to generate the examples. The standard ℓ_1 regularizer (far right) performed very poorly in terms of structure recovery. In fact, the ℓ_1-regularizer did not yield a single row of zeros in any of the experiments, underscoring one of the merits of using mixed-norm regularization in structured problems. Quantitatively, 96.3% and 94.5% of the zero rows of W were correctly identified when using FOBOS with ℓ_1/ℓ_2 and ℓ_1/ℓ_∞ regularization, respectively. In contrast, not one of the zero rows of W was identified correctly as an all zero row using pure ℓ_1 regularization.

The right plot in Fig. 11 shows the sparsity levels (fraction of non-zero weights) achieved by FOBOS as a function of the number of iterations of the algorithm. Each line represents a different synthetic experiment as λ is modified to give more or less sparsity to the solution vector $\boldsymbol{w}^\star$. The results demonstrate that FOBOS quickly selects the sparsity pattern of $\boldsymbol{w}^\star$, and the level of sparsity persists throughout its execution. We found this sparsity pattern common to all problems we tested, including mixed-norm problems. This is not particularly surprising, as Hale et al. (2007) recently gave an analysis showing that after a finite number of iterations, FOBOS-like algorithms attain the sparsity of the true solution $\boldsymbol{w}^\star$.

% Non-zero	ℓ_1 Test	ℓ_1/ℓ_2 Test	ℓ_1/ℓ_∞ Test
5	.43	**.29**	.40
10	.30	**.25**	.30
20	.26	**.22**	.26
40	.22	**.19**	.22

Table 2: LandSat classification error versus sparsity

% Non-zero	ℓ_1 Test	ℓ_1/ℓ_2 Test	ℓ_1/ℓ_∞ Test
5	.37	**.36**	.47
10	**.26**	**.26**	.31
20	**.15**	**.15**	.24
40	**.08**	**.08**	.16

Table 3: MNIST classification error versus sparsity

For comparison of the different regularization approaches, we report in Table 2 and Table 3 the test set error as a function of row sparsity of the learned matrix W. For the LandSat data, we see that using the block ℓ_1/ℓ_2 regularizer yields better performance for a given level of structural sparsity. However, on the MNIST data the ℓ_1 regularization and the ℓ_1/ℓ_2 achieve comparable performance for each level of structural sparsity. Moreover, for a given level of structural sparsity, the ℓ_1-regularized solution matrix W attains significantly higher overall sparsity, roughly 90% of the entries of each non-zero row are zero. The different performance on the different data sets might indicate that structural sparsity is effective only when the set of parameters indeed exhibit natural grouping.

For our final experiment, we show the power of FOBOS with mixed-norm regularization in the context of image compression. For this experiment, we represent each image as a set of patches where each patch is in turn represented as a 79 dimensional vector as described by Grangier and Bengio (2008). The goal is to jointly describe the set of patches by a single high-dimensional yet sparse set of dictionary features. Each of the dictionary terms is also in $\mathbb{R}^{79}$. Let $\boldsymbol{x}_j$ denote the j^{th} patch of an image with k patches to be compressed and $\boldsymbol{c}_i$ be the i^{th} dictionary vector from a dictionary of n vectors. The regularized objective is thus

$$\frac{1}{2}\sum_{j=1}^{k}\left\|\boldsymbol{x}_j-\sum_{i=1}^{n}w_{ij}\boldsymbol{c}_i\right\|_2^2+\lambda\sum_{i=1}^{n}\|\bar{\boldsymbol{w}}^i\|_q \ .$$

In our experiments, the number of dictionary vectors n was 1000 and the number of patches k was around 120 on average. We report results averaged over 100 different images. We experiment with the three settings for q we have used in prior experiments, namely $q \in \{1,2,\infty\}$. In Fig. 12 we report the average reconstruction error as a function of the fraction of dictionary vectors actually used. As one would expect, the mixed-norm regularizers (ℓ_1/ℓ_2 and ℓ_1/ℓ_∞) achieve lower reconstruction error as a function of dictionary sparsity than strict ℓ_1-regularization. The ℓ_1/ℓ_2-regularization also gives a slight, but significant, reconstruction improvement over ℓ_1/ℓ_∞-regularization. We hypothesize that this is related to the relative efficiency of ℓ_1/ℓ_∞ as a function of the geometry of the input space, as was theoretically discussed in Negahban and Wainwright (2008). Further investigation is required to shed more light into this type of phenomenon, and we leave it for future research.

8. Conclusions and Future Work

In this paper we analyzed a framework for online and batch convex optimization with a diverse class of regularization functions. We provided theoretical justification for a type of convex programming method we call

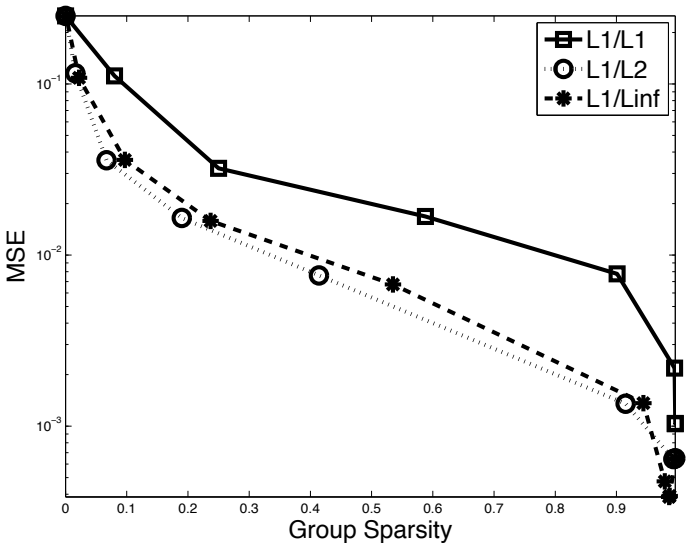


Figure 12: Image reconstruction error as a function of group sparsity.

FOBOS, which is known also as forward-backward splitting, iterative shrinkage and thresholding for the special case of ℓ_1-regularized problems, or SPARSA. Specifically, we described both offline convergence rates for arbitrary convex functions with regularization as well as regret bounds for online convex programming of regularized losses. Our derivation includes as a corollary the case of ℓ_1 regularization, which was concretely studied by Langford et al. (2008). Our approach provides a simple mechanism for solving online convex programs with many regularization functions, giving sparsity in parameters and different types of block or group regularization straightforwardly. Furthermore, the FOBOS framework is general and able to minimize any convex subdifferentialable function f so long as the forward looking step of Eq. (3) can be computed.

We have also provided a good deal of empirical evaluation of the method in comparison to other modern optimization methods for similar problems. Our practical experience suggests that for small to medium problems, SPARSA is effective and simple to implement (as opposed to more complicated coordinate descent methods), while for large scale problems, performing stochastic FOBOS is probably preferable. We have also shown that FOBOS is efficient for online learning with sparse data.

A few directions for further research suggest themselves, but we list here only two. The first is the question of whether we can modify the algorithm to work with arbitrary Bregman divergences of a function h instead of squared Euclidean distance, that is, we would like to form a generalized FOBOS update which is based on instantaneous optimization problems with Bregman divergences for convex differentiable h, where $B_h(\boldsymbol{u}, \boldsymbol{v}) = h(\boldsymbol{u}) - h(\boldsymbol{v}) - \langle \nabla h(\boldsymbol{v}), \boldsymbol{u} - \boldsymbol{v} \rangle$. We assume the generalized update would, loosely speaking, be analogous to nonlinear projected subgradient methods and the mirror descent (see, e.g., Beck and Teboulle 2003). This might allow us to give bounds for our algorithms in terms of other dual norms, such as ℓ_1/ℓ_∞ norms on the gradients or diameter of the space, rather than simply ℓ_2. We believe the attainment and rate of sparsity when using stochastic gradient information, as suggested by the discussion of Fig. 10, merits deeper investigation that will be fruitful and interesting.

Acknowledgments

We would like to thank Sam Roweis and Samy Bengio for helpful discussions and Shai Shalev-Shwartz and Tong Zhang for useful feedback on earlier drafts. We also would like to thank the anonymous reviewers and

editor Yoav Freund for their constructive comments and guidance. A large portion of John Duchi's work was performed at Google.

Appendix A. Online Regret Proofs

Proof of Theorem 6 Looking at Lemma 1, we immediately see that if $\|\partial f\|$ and $\|\partial r\|$ are bounded by G,

$$f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{w}^\star) + r(\boldsymbol{w}_{t+1}) - r(\boldsymbol{w}^\star) \leq \frac{1}{2\eta_t}\left(\|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{w}^\star\|^2\right) + \frac{7}{2}G^2\eta_t. \tag{36}$$

Now we use Eq. (36) to obtain that

$$\begin{aligned} R_{f+r}(T) &= \sum_{t=1}^{T}\left(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{w}^\star) + r(\boldsymbol{w}_t) - r(\boldsymbol{w}^\star)\right) + r(\boldsymbol{w}_{T+1}) - r(\boldsymbol{w}^\star) - r(\boldsymbol{w}_1) + r(\boldsymbol{w}^\star) \\ &\leq GD + \sum_{t=1}^{T}\frac{1}{2\eta_t}\left(\|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{w}^\star\|^2\right) + \frac{7G^2}{2}\sum_{t=1}^{T}\eta_t \end{aligned}$$

since $r(\boldsymbol{w}) \leq r(\mathbf{0}) + G\|\boldsymbol{w}\| \leq GD$. We can rewrite the above bound and see

$$\begin{aligned} R_{f+r}(T) &\leq GD + \frac{1}{2\eta_1}\|\boldsymbol{w}_1 - \boldsymbol{w}^\star\|^2 + \frac{1}{2}\sum_{t=2}^{T}\|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) + \frac{7G^2}{2}\sum_{t=1}^{T}\eta_t \\ &\leq GD + \frac{D^2}{2\eta_1} + \frac{D^2}{2}\sum_{t=2}^{T}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) + \frac{7G^2}{2}\sum_{t=1}^{T}\eta_t \ , \end{aligned}$$

where we used again the bound on the distance of each $\boldsymbol{w}_t$ to $\boldsymbol{w}^\star$ for the last inequality. Lastly, we use the fact that the sum $\frac{1}{\eta_1} + \sum_{t=2}^{T}(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}})$ telescopes and get that

$$R_{f+r}(T) \ \leq \ GD + \frac{D^2}{2\eta_T} + \frac{7G^2}{2}\sum_{t=1}^{T}\eta_t \ .$$

Setting $\eta_t = c/\sqrt{t}$ and recognizing that $\sum_{t=1}^{T}\eta_t \leq 2c\sqrt{T}$ concludes the proof. ■

Proof of Theorem 8 The proof builds straightforwardly on Theorem 1 from Hazan et al. (2006) and our proof of Theorem 6. We sum Eq. (17) from $t = 1$ to T and get

$$\begin{aligned} R_{f+r}(T) &\leq \sum_{t=1}^{T}\left(\left\langle \boldsymbol{g}_t^f + \boldsymbol{g}_t^r, \boldsymbol{w}_t - \boldsymbol{w}^\star\right\rangle - \frac{H}{2}\|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2\right) \\ &\leq 2GD + \frac{1}{2}\sum_{t=1}^{T-1}\left(\frac{1}{\eta_t}\|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2 - \frac{1}{\eta_t}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^\star\|^2 - H\|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2\right) + \frac{7G^2}{2}\sum_{t=1}^{T-1}\eta_t \\ &\leq 2GD + \frac{1}{2}\sum_{t=2}^{T-1}\|\boldsymbol{w}_t - \boldsymbol{w}^\star\|^2\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} - H\right) + \frac{1}{\eta_1}\|\boldsymbol{w}_1 - \boldsymbol{w}^\star\|^2 + \frac{7G^2}{2}\sum_{t=1}^{T-1}\eta_t. \end{aligned}$$

The second inequality follows from Eq. (36) and the third inequality from a rearrangement of the sum and removal of the negative term $(1/\eta_{T-1})\|\boldsymbol{w}_T - \boldsymbol{w}^\star\|^2$. Taking $\eta_t = \frac{1}{Ht}$, we see that $1/\eta_t - 1/\eta_{t-1} - H = Ht - H(t-1) - H = 0$, so we can bound the regret by

$$R_{f+r}(T) \leq 2GD + HD^2 + \frac{7G^2}{2}\sum_{t=1}^{T-1}\frac{1}{Ht} \leq 2GD + HD^2 + \frac{7G^2}{2H}(1 + \log T) = O\left(\frac{G^2}{H}\log T\right) \ .$$

■

Proof of Lemma 9 The triangle inequality implies that

$$\left\|\sum_{t=1}^{T} \partial f_t(\boldsymbol{w}_t)\right\| \leq \sum_{t=1}^{T} \|\partial f_t(\boldsymbol{w}_t)\| \leq TG \ .$$

Let $\boldsymbol{g}_t^\star \in \partial f_t(\boldsymbol{w}^\star)$ be such that

$$\mathbf{0} = \sum_{t=1}^{T} \boldsymbol{g}_t^\star + T\lambda \boldsymbol{w}^\star \in \sum_{t=1}^{T} \partial f_t(\boldsymbol{w}^\star) + T\partial r(\boldsymbol{w}^\star)$$

so $\|\boldsymbol{w}^\star\| = \|\sum_{t=1}^{T} \boldsymbol{g}_t^\star\|/(T\lambda) \leq G/\lambda$.

For the second part, assume that $\boldsymbol{w}_0 = \mathbf{0}$, and for our induction that $\boldsymbol{w}_t$ satisfies $\|\boldsymbol{w}_t\| \leq G/\lambda$. Then computing the FOBOS update from Eq. (20),

$$\|\boldsymbol{w}_{t+1}\| = \frac{\|\boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f\|}{1+\lambda\eta_t} \leq \frac{\|\boldsymbol{w}_t\| + \eta_t\|\boldsymbol{g}_t^f\|}{1+\lambda\eta_t} \leq \frac{G/\lambda + \eta_t G}{1+\lambda\eta_t} = \frac{G(1+\lambda\eta_t)}{\lambda(1+\lambda\eta_t)} \ .$$

■

Appendix B. Update for Berhu Regularization

Recalling the Berhu regularizer and combining it with Eq. (18) for one variable, we see that we want to minimize

$$\frac{1}{2}(w-v)^2 + \tilde{\lambda} b(w) = \frac{1}{2}(w-v)^2 + \tilde{\lambda}\left[|w|\,[\![|w| \leq \gamma]\!] + \frac{w^2+\gamma^2}{2\gamma}\,[\![|w| > \gamma]\!]\right].$$

First, if $|v| \leq \tilde{\lambda}$, then exactly reasoning to that for minimization of the ℓ_1-regularized minimization step implies that the optimal solution is $w = 0$.

When $|v| > \tilde{\lambda}$, there are two remaining cases to check. Let us assume without loss of generality that $v > \tilde{\lambda}$. It is immediate to verify that $w \geq 0$ at the optimum. Now, suppose that $v - \tilde{\lambda} \leq \gamma$. Taking $w = v - \tilde{\lambda} \leq \gamma$ (so that $w > 0$) gives us that $\partial b(w) = \{\tilde{\lambda}\}$. Thus, the subgradient set of our objective contains a single element, $w - v + \tilde{\lambda}\partial|w| = v - \tilde{\lambda} - v + \tilde{\lambda}1 = 0$. Therefore, when $v - \tilde{\lambda} \leq \gamma$ the optimal value of w is $v - \tilde{\lambda}$. The last case we need to examine is when $v - \tilde{\lambda} > \gamma$, which we as we show shortly puts the solution $w^\star$ in the ℓ_2^2 realm of $b(w)$. By choosing $w = \frac{v}{1+\frac{\tilde{\lambda}}{\gamma}}$ we get that,

$$w = \frac{v}{1+\frac{\tilde{\lambda}}{\gamma}} = \frac{v\gamma}{\gamma+\tilde{\lambda}} > \frac{(\gamma+\tilde{\lambda})\gamma}{\gamma+\tilde{\lambda}} = \gamma \ .$$

Therefore, $w > \gamma$ and thus w is in the ℓ_2^2 region of the Berhu penalty $b(w)$. Furthermore, for this choice of w the derivative of the penalty is

$$w - v + \tilde{\lambda}\frac{w}{\gamma} = \frac{v\gamma}{\gamma+\tilde{\lambda}} - v + \tilde{\lambda}\frac{v\gamma}{\gamma(\gamma+\tilde{\lambda})} = \frac{v\gamma}{\gamma+\tilde{\lambda}} - \frac{v(\gamma+\tilde{\lambda})}{\gamma+\tilde{\lambda}} + \tilde{\lambda}\frac{v}{\gamma+\tilde{\lambda}} = 0.$$

Combining the above results, inserting the conditions on the sign, and expanding $v = w_{t+\frac{1}{2},j}$ gives Eq. (26).

Appendix C. Fast Convergence Rate for Smooth Objectives

In this appendix, we describe an analysis of FOBOS which yields an $O(1/T)$ rate of convergence when f has Lipschitz-continuous gradient. Our analysis is by no means new. It is a distilled and simplified adaptation of the analysis of Nesterov (2007) to our setting.

Throughout the appendix we assume that $\nabla f(\boldsymbol{w})$ is Lipschitz continuous with a constant L, that is, $\|\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{v})\| \leq L\|\boldsymbol{w} - \boldsymbol{v}\|$. The fundamental theorem of calculus then readily implies that (Nesterov, 2004, Lemma 1.2.3)

$$|f(\boldsymbol{w}) - f(\boldsymbol{v}) - \langle \nabla f(\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v}\rangle| \leq \frac{L}{2}\|\boldsymbol{w} - \boldsymbol{v}\|^2 \ . \tag{37}$$

To see that Eq. (37) holds, add and subtract $\langle \nabla f(\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v}\rangle$ to note that

$$\begin{aligned} f(\boldsymbol{w}) - f(\boldsymbol{v}) &= \int_0^1 \langle \nabla f(\boldsymbol{v} + t(\boldsymbol{w} - \boldsymbol{v})), \boldsymbol{w} - \boldsymbol{v}\rangle \, dt \\ &= \langle \nabla f(\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v}\rangle + \int_0^1 \langle \nabla f(\boldsymbol{v} + t(\boldsymbol{w} - \boldsymbol{v})) - \nabla f(\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v}\rangle \, dt \end{aligned}$$

which, by using Cauchy-Shwartz inequality, yields

$$\begin{aligned} |f(\boldsymbol{w}) - f(\boldsymbol{v}) - \langle \nabla f(\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v}\rangle| \ &\leq \ \int_0^1 |\langle \nabla f(\boldsymbol{v} + t(\boldsymbol{w} - \boldsymbol{v})) - \nabla f(\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v}\rangle| \, dt \\ \leq \ \int_0^1 \|\nabla f(\boldsymbol{v} + t(\boldsymbol{w} - \boldsymbol{v})) - \nabla f(\boldsymbol{v})\| \|\boldsymbol{w} - \boldsymbol{v}\| dt \ &\leq \ \int_0^1 tL\|\boldsymbol{w} - \boldsymbol{v}\|^2 dt \ = \ \frac{L}{2}\|\boldsymbol{w} - \boldsymbol{v}\|^2 \ . \end{aligned}$$

For the remainder of this section, we assume that $f(\boldsymbol{w}) + r(\boldsymbol{w})$ is coercive, so that as $\|\boldsymbol{w}\| \to \infty$, $f(\boldsymbol{w}) + r(\boldsymbol{w}) \to \infty$. We thus have that the level sets of $f(\boldsymbol{w}) + r(\boldsymbol{w})$ are bounded: $\|\boldsymbol{w} - \boldsymbol{w}^\star\| \leq D$ for all $\boldsymbol{w}$ such that $f(\boldsymbol{w}) + r(\boldsymbol{w}) \leq f(\boldsymbol{0}) + r(\boldsymbol{0})$. Consider the "composite gradient mapping" (Nesterov, 2007)

$$m(\boldsymbol{v}, \boldsymbol{w}) = f(\boldsymbol{v}) + \langle \nabla f(\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v}\rangle + \frac{L}{2}\|\boldsymbol{w} - \boldsymbol{v}\|^2 + r(\boldsymbol{w}) \ . \tag{38}$$

Before proceeding with the proof of fast convergence rate, we would like to underscore the equivalence of the FOBOS update and the composite gradient mapping. Formally, minimizing $m(\boldsymbol{v}, \boldsymbol{w})$ with respect to $\boldsymbol{w}$ is completely equivalent to taking a FOBOS step with $\eta = 1/L$ and $\boldsymbol{v} = \boldsymbol{w}_t$. To obtain the FOBOS update from Eq. (38) we simply need to divide $m(\boldsymbol{v}, \boldsymbol{w})$ by $L = 1/\eta$, omit terms that solely depend on $\boldsymbol{v} = \boldsymbol{w}_t$, and use the fact that $\boldsymbol{w}_{t+\frac{1}{2}} = \boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f = \boldsymbol{v} - \nabla f(\boldsymbol{v})/L$.

For notational convenience, let $\phi(\boldsymbol{w}) = f(\boldsymbol{w}) + r(\boldsymbol{w})$. Denote by $\boldsymbol{w}^+$ the vector minimizing $m(\boldsymbol{v}, \boldsymbol{w})$. Then from Eq. (37) we get that

$$\phi(\boldsymbol{w}^+) = f(\boldsymbol{w}^+) + r(\boldsymbol{w}^+) \leq f(\boldsymbol{v}) + \left\langle \nabla f(\boldsymbol{v}), \boldsymbol{w}^+ - \boldsymbol{v}\right\rangle + \frac{L}{2}\|\boldsymbol{w}^+ - \boldsymbol{v}\|^2 + r(\boldsymbol{w}^+) = \inf_{\boldsymbol{w}} m(\boldsymbol{v}, \boldsymbol{w}). \tag{39}$$

Further, because $f(\boldsymbol{v}) + \langle \nabla f(\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v}\rangle \leq f(\boldsymbol{w})$ for all $\boldsymbol{w}$, we have

$$\inf_{\boldsymbol{w}} m(\boldsymbol{v}, \boldsymbol{w}) \leq \inf_{\boldsymbol{w}} \left[f(\boldsymbol{w}) + \frac{L}{2}\|\boldsymbol{w} - \boldsymbol{v}\|^2 + r(\boldsymbol{w}) \right] = \inf_{\boldsymbol{w}} \left[\phi(\boldsymbol{w}) + \frac{L}{2}\|\boldsymbol{w} - \boldsymbol{v}\|^2 \right] \ . \tag{40}$$

Now we consider the change in function value from $\boldsymbol{w}_t$ to $\boldsymbol{w}_{t+1}$ for the FOBOS update with $\eta = 1/L$. To do this, we take an arbitrary optimal point $\boldsymbol{w}^\star$ and restrict $\boldsymbol{w}_{t+1}$ to lie on the line between $\boldsymbol{w}_t$ and $\boldsymbol{w}^\star$, which constrains the set of infimum values above and allows us to carefully control them. With this construction,

along with Eqs. (39) and (40) we get that

$$\begin{aligned}
\phi(\boldsymbol{w}_{t+1}) &\leq \inf_{\boldsymbol{w}} \left[\phi(\boldsymbol{w}) + \frac{L}{2} \|\boldsymbol{w} - \boldsymbol{w}_t\|^2 \right] \\
&\leq \inf_{\alpha \in [0,1]} \left[\phi(\alpha \boldsymbol{w}^\star + (1-\alpha)\boldsymbol{w}_t) + \frac{L}{2} \|\alpha \boldsymbol{w}^\star + (1-\alpha)\boldsymbol{w}_t - \boldsymbol{w}_t\|^2 \right] \\
&\leq \inf_{\alpha \in [0,1]} \left[\alpha \phi(\boldsymbol{w}^\star) + (1-\alpha)\phi(\boldsymbol{w}_t) + \frac{\alpha^2 L}{2} \|\boldsymbol{w}^\star - \boldsymbol{w}_t\|^2 \right] . \qquad (41)
\end{aligned}$$

The bound in Eq. (41) follows due to the convexity of ϕ. One immediate consequence of Eq. (41) is that $\phi(\boldsymbol{w}_{t+1}) \leq \phi(\boldsymbol{w}_t)$, since at $\alpha = 0$ we obtain the same objective for ϕ. Thus, all iterates of the method satisfy $\|\boldsymbol{w}_t - \boldsymbol{w}^\star\| \leq D$. We therefore can distill the bound to be

$$\phi(\boldsymbol{w}_{t+1}) \leq \inf_{\alpha \in [0,1]} \left[\phi(\boldsymbol{w}_t) + \alpha \left(\phi(\boldsymbol{w}^\star) - \phi(\boldsymbol{w}_t) \right) + \alpha^2 \frac{LD^2}{2} \right] .$$

The argument of the infimum of the equation above is a quadratic equation in α. We need to analyze two possible cases for the optimal solution. In the first case when $\phi(\boldsymbol{w}_t) - \phi(\boldsymbol{w}^\star) > LD^2$, the optimal value of α is 1 and $\phi(\boldsymbol{w}_{t+1}) \leq \phi(\boldsymbol{w}^\star) + LD^2/2$. Therefore, we will never encounter again this case in future iterations. The second occurs when $\phi(\boldsymbol{w}_t) - \phi(\boldsymbol{w}^\star) \leq LD^2$, so we have $\alpha = (\phi(\boldsymbol{w}_t) - \phi(\boldsymbol{w}^\star))/LD^2 \in [0,1]$, which yields

$$\phi(\boldsymbol{w}_{t+1}) \leq \phi(\boldsymbol{w}_t) - \frac{\left(\phi(\boldsymbol{w}_t) - \phi(\boldsymbol{w}^\star)\right)^2}{2LD^2} . \qquad (42)$$

To obtain the form of the convergence rate let us define the inverse residual value $\rho_t = 1/(\phi(\boldsymbol{w}_t) - \phi(\boldsymbol{w}^\star))$. By analysing the rate at which ρ_t tends to infinity we obtain our desired convergence rate. From the definition of ρ_t and the bound of Eq. (42) we get that

$$\begin{aligned}
\rho_{t+1} - \rho_t &= \frac{1}{\phi(\boldsymbol{w}_{t+1}) - \phi(\boldsymbol{w}^\star)} - \frac{1}{\phi(\boldsymbol{w}_t) - \phi(\boldsymbol{w}^\star)} = \frac{\phi(\boldsymbol{w}_t) - \phi(\boldsymbol{w}^\star) - \phi(\boldsymbol{w}_{t+1}) + \phi(\boldsymbol{w}^\star)}{(\phi(\boldsymbol{w}_{t+1}) - \phi(\boldsymbol{w}^\star))(\phi(\boldsymbol{w}_t) - \phi(\boldsymbol{w}^\star))} \\
&= \rho_t \rho_{t+1} (\phi(\boldsymbol{w}_t) - \phi(\boldsymbol{w}_{t+1})) \geq \rho_t \rho_{t+1} \frac{\left(\phi(\boldsymbol{w}_t) - \phi(\boldsymbol{w}^\star)\right)^2}{2LD^2} = \frac{\rho_t \rho_{t+1}}{2\rho_t^2 LD^2} \geq \frac{1}{LD^2} ,
\end{aligned}$$

where the last inequality is due to the fact that $\rho_{t+1} \geq \rho_t$ and therefore $\rho_t \rho_{t+1}/\rho_t^2 \geq 1$. Summing the differences $\rho_{t+1} - \rho_t$ from $t = 0$ through $T-1$, we get $\rho_T \geq T/2LD^2$. Thus, for $t \geq 1$ we have

$$\phi(\boldsymbol{w}_t) - \phi(\boldsymbol{w}^\star) = 1/\rho_t \leq \frac{2LD^2}{t} .$$

To recap, by setting $\eta = 1/L$ while relaying on the fact that f has Lipschitz continuous gradient with constant L, we obtain a $1/T$ rate of convergence

$$f(\boldsymbol{w}_T) + r(\boldsymbol{w}_T) \leq f(\boldsymbol{w}^\star) + r(\boldsymbol{w}^\star) + \frac{2LD^2}{T} .$$

References

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31:167–175, 2003.

D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

J. Blitzer, M. Dredze, and F. Pereira. Biographies, Bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In *Association for Computational Linguistics*, 2007.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

G. Chen and R. T. Rockafellar. Convergence rates in forward-backward splitting. *SIAM Journal on Optimization*, 7(2), 1997.

P. Combettes and V. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4(4):1168–1200, 2005.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communication on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

I. Daubechies, M. Fornasier, and I. Loris. Accelerated projected gradient method for linear inverse problems with sparsity constraints. *Fourier Analysis and Applications*, 14(5):764–792, 2008.

D. L. Donoho. De-noising via soft thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

D. Grangier and S. Bengio. A discriminative kernel-based model to rank images from text queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1371–1384, 2008.

E. Hale, W. Yin, and Y. Zhang. A fixed-point continuation method for ℓ_1-regularized minimization with applications to compressed sensing. Technical Report TR07-07, Rice University Department of Computational and Applied Mathematics, July 2007.

E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2006.

K. Koh, S.J. Kim, and S. Boyd. An interior-point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. In *Advances in Neural Information Processing Systems 22*, 2008.

P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16:964–979, 1979.

L. Meier, S. van de Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society Series B*, 70(1):53–71, 2008.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34:1436–1462, 2006.

S. Negahban and M. Wainwright. Phase transitions for high-dimensional joint support recovery. In *Advances in Neural Information Processing Systems 22*, 2008.

Y. Nesterov. *Introductory Lectures on Convex Optimization*. Kluwer Academic Publishers, 2004.

Y. Nesterov. Gradient methods for minimizing composite objective function. Technical Report 76, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain (UCL), 2007.

G. Obozinski, B. Taskar, and M. Jordan. Joint covariate selection for grouped classification. Technical Report 743, Dept. of Statistics, University of California Berkeley, 2007.

G. Obozinski, M. Wainwright, and M. Jordan. High-dimensional union support recovery in multivariate regression. In *Advances in Neural Information Processing Systems 22*, 2008.

Art Owen. A robust hybrid of lasso and ridge regression. Technical report, Stanford University, 2006.

A. Quattoni, X. Carreras, M. Collins, and T. Darrell. An efficient projection for L1,infinity regularization. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

M. Schmidt, E. van den Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: a limited-memory projected quasi-Newton method. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, 2009.

S. Shalev-Shwartz and Y. Singer. Logarithmic regret algorithms for strongly convex repeated games. Technical report, The Hebrew University, 2007. Available at http://www.cs.huji.ac.il/∼shais.

S. Shalev-Shwartz and A. Tewari. Stochastic methods for ℓ_1-regularized loss minimization. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

D. Spiegelhalter and C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

P. Tseng. A modified forward backward splitting method for maximal monotone mappings. *SIAM Journal on Control and Optimization*, 38:431–446, 2000.

P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming Series B*, 117:387–423, 2007.

S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

P. Zhao and B. Yu. On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7: 2541–2567, 2006.

P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties. Technical Report 703, Statistics Department, University of California Berkeley, 2006.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

Journal of Machine Learning Research 10 (2009) 2935-2962 Submitted 11/09; Published 12/09

A Survey of Accuracy Evaluation Metrics of Recommendation Tasks

Asela Gunawardana ASELAG@MICROSOFT.COM
Microsoft Research
One Microsoft Way
Redmond, WA 98052-6399, USA

Guy Shani SHANIGU@BGU.AC.IL
Information Systems Engineering
Ben Gurion University
Beer Sheva, Israel

Editor: Lyle Ungar

Abstract

Recommender systems are now popular both commercially and in the research community, where many algorithms have been suggested for providing recommendations. These algorithms typically perform differently in various domains and tasks. Therefore, it is important from the research perspective, as well as from a practical view, to be able to decide on an algorithm that matches the domain and the task of interest. The standard way to make such decisions is by comparing a number of algorithms offline using some evaluation metric. Indeed, many evaluation metrics have been suggested for comparing recommendation algorithms. The decision on the proper evaluation metric is often critical, as each metric may favor a different algorithm. In this paper we review the proper construction of offline experiments for deciding on the most appropriate algorithm. We discuss three important tasks of recommender systems, and classify a set of appropriate well known evaluation metrics for each task. We demonstrate how using an improper evaluation metric can lead to the selection of an improper algorithm for the task of interest. We also discuss other important considerations when designing offline experiments.

Keywords: recommender systems, collaborative filtering, statistical analysis, comparative studies

1. Introduction

Recommender systems can now be found in many modern applications that expose the user to a huge collections of items. Such systems typically provide the user with a list of recommended items they might prefer, or supply guesses of how much the user might prefer each item. These systems help users to decide on appropriate items, and ease the task of finding preferred items in the collection.

For example, the DVD rental provider Netflix[1] displays predicted ratings for every displayed movie in order to help the user decide which movie to rent. The online book retailer Amazon[2] provides average user ratings for displayed books, and a list of other books that are bought by users who buy a specific book. Microsoft provides many free downloads for users, such as bug fixes, products and so forth. When a user downloads some software, the system presents a list

1. This can be found at `www.netflix.com`.
2. This can be found at `www.amazon.com`.

of additional items that are downloaded together. All these systems are typically categorized as recommender systems, even though they provide diverse services.

In the past decade, there has been a vast amount of research in the field of recommender systems, mostly focusing on designing new algorithms for recommendations. An application designer who wishes to add a recommendation system to her application has a large variety of algorithms at her disposal, and must make a decision about the most appropriate algorithm for her application. Typically, such decisions are based on offline experiments, comparing the performance of a number of candidate algorithms over real data. The designer can then select the best performing algorithm, given structural constraints. Furthermore, most researchers who suggest new recommendation algorithms also compare the performance of their new algorithm to a set of existing approaches. Such evaluations are typically performed by applying some evaluation metric that provides a ranking of the candidate algorithms (usually using numeric scores).

Many evaluation metrics have been used to rank recommendation algorithms, some measuring similar features, but some measuring drastically different quantities. For example, methods such as the Root of the Mean Square Error (RMSE) measure the distance between predicted preferences and true preferences over items, while the Recall method computes the portion of favored items that were suggested. Clearly, it is unlikely that a single algorithm would outperform all others over all possible methods.

Therefore, we should expect different metrics to provide different rankings of algorithms. As such, selecting the proper evaluation metric to use has a crucial influence on the selection of the recommender system algorithm that will be selected for deployment. This survey reviews existing evaluation metrics, suggesting an approach for deciding which evaluation metric is most appropriate for a given application.

We categorize previously suggested recommender systems into three major groups, each corresponding to a different *task*. The first obvious task is to recommend a set of good (interesting, useful) items to the user. In this task it is assumed that all good items are interchangeable. A second, less discussed, although highly important task is utility optimization. For example, many e-commerce websites use a recommender system, hoping to increase their revenues. In this case, the task is to present a set of recommendations that will optimize the retailer revenue. Finally, a very common task is the prediction of user opinion (e.g., rating) over a set of items. While this may not be an explicit act of recommendation, much research in recommender systems focuses on this task, and so we address it here.

For each such task we review a family of common evaluation metrics that measure the performance of algorithms on that task. We discuss the properties of each such metric, and why it is most appropriate for a given task.

In some cases, applying incorrect evaluation metrics may result in selecting an inappropriate algorithm. We demonstrate this by experimenting with a wide collection of data sets, comparing a number of algorithms using various evaluation metrics, showing that the metrics rank the algorithms differently.

We also discuss the proper design of an offline experiment, explaining how the data should be split, which measurements should be taken, how to determine if differences in performance are statistically significant, and so forth. We also describe a few common pitfalls that may produce results that are not statistically sound.

The paper is structured as follows: we begin with some necessary background on recommender approaches (Section 2). We categorize recommender systems into a set of three tasks in Section 3.

We then discuss evaluation protocols, including online experimentation, offline testing, and statistical significance testing of results in Section 4. We proceed to review a set of existing evaluation metrics, mapping them to the appropriate task (Section 5). We then provide (Section 6) some examples of applying different metrics to a set of algorithms, resulting in questionable rankings of these algorithms when inappropriate measures are used. Following this, we discuss some additional relevant topics that arise (Section 7) and some related work (Section 8), and then conclude (Section 9).

2. Algorithmic Approaches

There are two dominant approaches for computing recommendations for the *active user*—the user that is currently interacting with the application and the recommender system. First, the *collaborative filtering* approach (Breese et al., 1998) assumes that users who agreed on preferred items in the past will tend to agree in the future too. Many such methods rely on a matrix of user-item ratings to predict unknown matrix entries, and thus to decide which items to recommend.

A simple approach in this family (Konstan et al., 2006), commonly referred to as *user based collaborative filtering*, identifies a neighborhood of users that are similar to the *active user*. This set of neighbors is based on the similarity of observed preferences between these users and the active user. Then, items that were preferred by users in the neighborhood are recommended to the active user. Another approach (Linden et al., 2003), known as *item based collaborative filtering* recommends items also prefered by users that prefer a particular *active item* to other users that also prefer that active item. In collaborative filtering approaches, the system only has access to the item and user identifiers, and no additional information over items or users is used. For example, websites that present recommendations titled "users who preferred this item also prefer" typically use some type of collaborative filtering algorithm.

A second popular approach is the *content-based* recommendation. In this approach, the system has access to a set of item features. The system then learns the user preferences over features, and uses these computed preferences to recommend new items with similar features. Such recommendations are typically titled "similar items". User's features, if available, such as demographics (e.g., gender, age, geographic location) can also provide valuable information.

Each approach has advantages and disadvantages, and a multitude of algorithms from each family, as well as a number of hybrid approaches have been suggested. This paper, though, makes no distinction between the underlying recommendation algorithms when evaluating their performance. Just as users should not need to take into account the details of the underlying algorithm when using the resulting recommendations, it is inappropriate to select different evaluation metrics for different recommendation approaches. In fact, doing so would make it difficult to decide which approach to employ in a particular application.

3. Recommender Systems Tasks

Providing a single definition for recommender systems is difficult, mainly because systems with different objectives and behaviors are grouped together under that name. Below, we categorize recommender systems into three classes, based on the recommendation task that they are designed for McNee et al. (2006). In fact, there have been several previous attempts to classify existing recommenders (see, e.g., Montaner et al. 2003 and Schafer et al. 1999). We, however, are interested

in the proper evaluation of such algorithms, and our classification is derived from that goal. While there may be recommender systems that do not fit well into the classes that we suggest, we believe that the vast majority of the recommender systems attempt to achieve one of these tasks, and can thus be classified as we suggest.

3.1 Recommending Good Items

Perhaps the most common task of recommendation engines is to recommend good items to users (Herlocker et al., 2004; McNee et al., 2006). Such systems typically present a list of items that the user is predicted to prefer. The user can then select (add to the shopping basket, view, ...) one or more of the suggested items. There are many examples of such systems. In the Amazon website, for instance, when the user is looking at an item, the system presents below a list of other items that the user may be interested in. Another example can be found in Netflix—when a user adds a movie to her queue, the system displays a list of other recommended movies that the user may want to add to the queue too. There are several considerations when creating recommendation lists. We identify below two sub-tasks that comply with different requirements.

3.1.1 RECOMMENDING SOME GOOD ITEMS

In this sub-task, we make the assumption that there is a large number of good items that may appeal to the user, and the user does not have enough resources (time, money) to select all items. In this case we can only present a part of the preferred item set. Thus, it is likely that many preferred items will be missing from the list. In this sub-task, it is more important not to present any disliked item than to find all the good items.

This is typically the case in recommender systems that suggest media items, such as movies, books, or news items. In all these cases the number of alternatives is huge, and the user cannot possibly watch all the recommended movies, or read all the relevant books.

3.1.2 RECOMMENDING ALL GOOD ITEMS

A less popular case is when the system should recommend all important items. Examples of such systems are recommenders that predict which scientific papers should be cited, or legal databases (Herlocker et al., 2004; McNee et al., 2006), where it is important not to overlook any possible case. In this sub-task, the system can present longer lists of items, trying to avoid missing a relevant item.

3.2 Optimizing Utility

With the rise of e-commerce websites, another recommendation task became highly important—maximizing the profits of the website. Online retailers are willing to invest in recommender systems hoping to increase their revenue. There are many ways by which a recommender system can increase revenue. The simplest way is through cross-selling; by suggesting additional items to the users, we increase the probability that the user will buy more than he originally intended. In an online news provider, where most revenue comes from display advertisements, the system can increase profit by keeping the users in the website for longer time periods, as the performance of an advertising campaign is often measured in terms of "x-minute reach," which is the number of consumers in a particular market that are exposed to the ad for x minutes. In such cases, it is in the best interest of the system to suggest items in order to lengthen the session. In a subscription

service, where revenue comes from users paying a regular subscription, the goal may be to allow users to easily reach items of interest. In this case, the system should suggest items such that the user reaches items of interest with minimal effort.

The utility function to be optimized can be more complicated, and in particular, may be a function of the entire set of recommendations and their presentation to the user. For example, in pay-per-click search advertising, the system must recommend advertisements to be displayed on search results pages. Each advertiser bids a fixed amount that is paid only when the user clicks on their ad. If we wish to optimize the expected system profit, both the bids and the probability that the user will click on each ad must be taken into account. This probability depends on the relevance of each ad to the user and the placement of the different ads on the page. Since the different ads displayed compete for the user's attention, the utility function depends on the entire set of ads displayed, and is not additive over the set (Gunawardana and Meek, 2008).

In all of these cases, it may be suboptimal to suggest items based solely on their predicted rating. While it is certainly beneficial to recommend relevant items, other considerations are also important. For example, in the e-commerce scenario, given two items that the system perceives as equally relevant, suggesting the item with the higher profit can further increase revenue. In the online news agency case, recommending longer stories may be beneficial, because reading them will keep the user in the website longer. In the subscription service, recommending items that are harder for the user to reach without the recommender system may be beneficial.

Another common practice of recommendation systems is to suggest recommendations that provide the most "value" to the user. For example, recommending popular items can be redundant, as the user is probably already familiar with them. A recommendation of a preferred, yet unknown item can provide a much higher value for the user.

Such approaches can be viewed as instances of providing recommendations that maximize some utility function that assigns a value to each recommendation. Defining the correct utility function for a given application can be difficult (Braziunas and Boutilier, 2005), and typically system designers make simplifying assumptions about the user utility function. In the e-commerce case the utility function is typically the profit resulting from recommending an item, and in the news scenario the utility can be the expected time for reading a news item, but these choices ignore the effect of the resulting recommendations on long-term profits. When we are interested in novel recommendations, the utility can be the log of the inverse popularity of an item, modeling the amount of new information in a recommended item (Shani et al., 2005), but this ignores other aspects of user-utility such as the diversity of recommendations.

In fact, it is possible to view many recommendation tasks, such as providing novel or serendipitious recommendations as maximizing some utility function. Also, the "recommend good items" of the previous section can be considered as optimizing for a utility function assigning a value of 1 to each successful recommendation. In this paper, due to the popularity of the former task, we choose to keep the two tasks distinct.

3.3 Predicting Ratings

In some cases, a system is required to predict the user ratings over a given set of items. For example, in the Netflix website, when the user is browsing the list of new releases, the system assigns a predicted rating for each movie. In CNET,[3] a website offering electronic product reviews, users can

3. This can be found at `www.cnet.com`.

search for, say, laptops that cost between $400 and $700. The system adds to some laptops in the list an automatically computed rating, based on the laptop features.

It is arguable whether this task is indeed a recommendation task. However, many researchers in the recommendations system community attempting to find good algorithms for this task. Examples include the Netflix competition, which was warmly embraced by the research community, and the numerous papers on predicting ratings on the Netflix or MovieLens[4] data sets.

While such systems do not provide lists of recommended items, predicting that the user will rate an item highly can be considered an act of recommendation. Furthermore, one can view a predicted high rating as a recommendation to use the item, and a predicted low rating as a recommendation to avoid the item. Indeed, it is common practice to use predicted ratings to generate a list of recommendations. Below, we will present several arguments of cases where this common practice may be undesirable.

4. Evaluation Protocols

We now discuss an experimental protocol for evaluating and choosing recommendation algorithms. We review several requirements to ensure that the results of the experiments are statistically sound. We also describe several common pitfalls in such experimental settings. This section reviews the evaluation protocols in related areas such as machine learning and information retrieval, highlighting practices relevant to evaluating recommendation systems. The reader is referred to publications in these fields for more detailed discussions (Salzberg, 1997; Demšar, 2006; Voorhees, 2002a).

We begin by discussing online experiments, which can measure the real performance of the system. We then argue that offline experiments are also crucial, because online experiments are costly in many cases. Therefore, the bulk of the section discusses the offline experimental setting in detail.

4.1 Online Evaluation

In the recommendation and utility optimization tasks, the designer of the system wishes to influence the behavior of users. We are therefore interested in measuring the change in user behavior when interacting with different recommendation systems. For example, if users of one system follow the recommendations more often (in the case of the "recommend good items" task), or if the utility gathered from users of one system exceeds utility gathered from users of the other system (in the utility optimization task), then we can conclude that one system is superior to the other, all else being equal. In the case of ratings prediction tasks, the goal is to provide information to support user browsing and search. Once again, the value of such predictions can depend on a variety of factors such as the user's intent (e.g., how specific their information needs are, how much novelty vs. how much risk they are seeking), the user's context (e.g., what items they are already familiar with, how much they trust the system), and the interface through which the predictions are presented.

For this reason, many real world systems employ an online testing system (Kohavi et al., 2009), where multiple algorithms can be compared. Typically, such systems redirect a small percentage of the traffic to each different recommendation engine, and record the users interactions with the different systems. There are a few considerations that must be made when running such tests. For example, it is important to sample (redirect) users randomly, so that the comparisons between

4. This can be found at `www.movielens.org`.

alternatives are fair. It is also important to single out the different aspects of the recommenders. For example, if we care about algorithmic accuracy, it is important to keep the user interface fixed. On the other hand, if we wish to focus on a better user interface, it is best to keep the underlying algorithm fixed.

However, in a multitude of cases, such experiments are very costly, since creating online testing systems may require much effort. Furthermore, we would like to evaluate our algorithms before presenting their results to the users, in order to avoid a negative user experience for the test users. For example, a test system that provides irrelevant recommendations, may discourage the test users from using the real system ever again. Finally, designers that wish to add a recommendation system to their application before its deployment do not have an opportunity to run such tests.

For these reasons, it is important to be able to evaluate the performance of algorithms in an offline setting, assuming that the results of these offline tests correlate well with the online behavior of users.

4.2 Offline Experimental Setup

As described above, the goal of the offline evaluation is to filter algorithms so that only the most promising need undergo expensive online tests. Thus, the data used for the offline evaluation should match as closely as possible the data the designer expects the recommender system to face when deployed online. Care must be exercised to ensure that there is no bias in the distributions of users, items and ratings selected. For example, in cases where data from an existing system (perhaps a system without a recommender) is available, the experimenter may be tempted to pre-filter the data by excluding items or users with low counts, in order to reduce the costs of experimentation. In doing so, the experimenter should be mindful that this involves a trade-off, since this introduces a systematic bias in the data. If necessary, randomly sampling users and items may be a preferable method for reducing data, although this can also introduce other biases into the experiment (e.g., this could tend to favor algorithms that work better with more sparse data).

In order to evaluate algorithms offline, it is necessary to simulate the online process where the system makes predictions or recommendations, and the user corrects the predictions or uses the recommendations. This is usually done by recording historical user data, and then hiding some of these interactions in order to simulate the knowledge of how a user will rate an item, or which recommendations a user will act upon.

There are a number of ways to choose the ratings/selected items to be hidden. Once again, it is preferable that this choice be done in a manner that simulates the target application as closely as possible. We discuss these concerns explicitly for the case of selecting used items for hiding in the evaluation of recommendation tasks, and note that the same considerations apply when selecting ratings to hide for evaluation of ratings prediction tasks.

Our goal is to simulate sets of past user selections that are representative of what the system will face when deployed. Ideally, if we have access to time-stamps for user selections, we can randomly sample test users, randomly sample a time just prior to a user action, hide all selections (of all users) after that instant, and then attempt to recommend items to that user. This protocol requires changing the set of given information prior to each recommendation, which can be computationally quite expensive. A cheaper alternative is to sample a set of test users, then sample a single test time, and hide all items after the sampled test time for each test user. This simulates a situation where the recommender system is "trained" as of the test time, and then makes recommendations without

taking into account any new data that arrives after the test time. Another alternative is to sample a test time for each test user, and hide the test user's items after that time, without maintaining time consistency across users. This effectively assumes that it is the sequence in which items are selected, and not the absolute times when they are selected that is important. A final alternative is to ignore time; We sample a set of test users, then sample the number n_a of items to hide for each user a, then sample n_a items to hide. This assumes that the temporal aspects of user selections are unimportant. All three of the latter alternatives partition the data into a single training set and single test set. It is important to select an alternative that is most appropriate for the domain and task of interest, rather than the most convenient one.

A common protocol used in many research papers is to use a fixed number of known items or a fixed number of hidden items per test user (so called "given n" or "all but n" protocols). This protocol is useful for diagnosing algorithms and identifying in which cases they work best. However, when we wish to make decisions on the algorithm that we will use in our application, we must ask ourselves whether we are truly interested in presenting recommendations for users who have rated exactly n items, or are expected to rate exactly n items more. If that is not the case, then results computed using these protocol have biases that make them difficult to use in predicting the outcome of using the algorithms online.

The evaluation protocol we suggest above generates a test set (Duda and Hart, 1973) which is used to obtain held-out estimates for algorithm performance, using performance measures which we discuss below. Another popular alternative is to use cross-validation (Stone, 1974), where the data is divided into a number of partitions, and each partition in turn is used as a test set. The advantages of the cross-validation approach are to allow the use of more data in ranking algorithms, and to take into account the effect of training set variation. In the case of recommender systems, the held-out approach usually yields enough data to make reliable decisions. Furthermore, in real systems, the problem of variation in training data is avoided by evaluating systems trained on the historical data specific to the task at hand. In addition, there is a risk that since the results on the different data partitions are not independent of each other, pooling the results across partitions for ranking algorithms can lead to statistically unjustified decisions (Bengio and Grandvalet, 2004).

4.3 Making Reliable Choices

When choosing between algorithms, it is important that we can be confidant that the algorithm that we choose will also be a good choice for the yet unseen data the system will be faced with in the future. As we explain above, we should exercise caution in choosing the data so that it would be most similar to the online application. Still, there is a possibility that the algorithm that performed best on this test set did so because the test set was fortuitously suitable for that algorithm. To reduce the possibility of such statistical mishaps, we must perform significance testing on the results.

Typically we compute a significance level or p-value—the probability that the obtained results were due to luck. Generally, we will reject the null hypothesis that algorithm A is no better than algorithm B if the p-value is above 0.05 (or below 95% confidence). That is, if the probability that the observed ranking is achieved by chance exceeds 0.05. More stringent significance levels (e.g., 0.01 or even lower) can be used in cases where the cost of making the wrong choice is higher.

In order to perform a significance test that algorithm A is indeed better than algorithm B, we require the results of several independent experiments comparing A and B. The protocol we have chosen in generating our test data ensures that we will have this set of results. Assuming that test

users are drawn independently from some population, the performance measures of the algorithms for each test user give us the independent comparisons we need. However, when recommendations or predictions of multiple items are made to the same user, it is unlikely that the resulting per-item performance metrics are independent. Therefore, it is better to compare algorithms on a per-user case. Approaches for use when users have not been sampled independently also exist, and attempt to directly model these dependencies (see, e.g., Larocque et al. 2007). Care should be exercised when using such methods, as it can be difficult to verify that the modeling assumptions that they depend on hold in practice.

Given such paired per-user performance measures for algorithms A and B the simplest test of significance is the sign test (Demšar, 2006). In this test, we count the number of users for whom algorithm A outperforms algorithm B (n_A) and the number of users for whom algorithm B outperforms algorithm A (n_B). The probability that A is not truly better than B is estimated as the probability of at least n_A out of $n_A + n_B$ 0.5-probability Binomial trials succeeding (that is, n_A out of $n_A + n_B$ fair coin-flips coming up "heads").

$$pr(successes \geq n_A | A = B) = 0.5^{n_A+n_B} \sum_{k=n_A}^{n_a+n_B} \frac{(n_A+n_B)!}{k!(n_i+n_B-k)!}.$$

The sign test is an attractive choice due to its simplicity, and lack of assumptions over the distribution of cases. Still this test may lead to mislabeling of significant results as insignificant when the number of test points is small. In these cases, the more sophisticated Wilcoxon signed rank test can be used (Demšar, 2006). As mentioned in Section 4.2, cross-validation can be used to increase the amount of data, and thus the significance of results, but in this case the results obtained on the cross-validated test sets are no longer independent, and care must be exercised to ensure that our decisions account for this (Bengio and Grandvalet, 2004). Also, model-based approaches (e.g., Goutte and Gaussier, 2005) may be useful when the amount of data is small, but once again, care must be exercised to ensure that the model assumptions are reasonable for the application at hand.

Another important consideration is the effect of evaluating multiple versions of algorithms. For example, an experimenter might try out several variants of a novel recommender algorithm and compare them to a baseline algorithm until they find one that passes a sign test at the $p = 0.05$ level and therefore infer that their algorithm improves upon the baseline with 95% confidence. However, this is not a valid inference. Suppose the experimenter evaluated ten different variants all of which are statistically the same as the baseline. If the probability that any one of these trials passes the sign test mistakenly is $p = 0.05$, the probability that at least one of the ten trials passes the sign test mistakenly is $1-(1-0.05)^{20} = 0.40$. This risk is colloquially known as "tuning to the test set" and can be avoided by separating the test set users into two groups—a development (or tuning) set, and an evaluation set. The choice of algorithm is done based on the development test, and the validity of the choice is measured by running a significance test on the evaluation set.

A similar concern exists when ranking a number of algorithms, but is more difficult to circumvent. Suppose the best of $N+1$ algorithms is chosen on the development test set. We can have a confidence $1-p$ that the chosen algorithm is indeed the best, if it outperforms the N other algorithms on the evaluation set with significance $1-(1-p)^{1/N}$. This is known as the Bonferroni correction, and should be used when pair-wise significant tests are used multiple times. Alternatively, the Friedman test for ranking can be used (Demšar, 2006).

5. Evaluating Tasks

An application designer that wishes to employ a recommendation system typically knows the purpose of the system, and can map it into one of the tasks defined above—recommendation, utility optimization, and ratings prediction. Given such a mapping, the designer must now decide which evaluation metric to use in order to rank a set of candidate recommendation algorithms. It is important that the metric match the task, to avoid an inappropriate ranking of the candidates.

Below we provide an overview of a large number of evaluation metrics that have been suggested in the recommendation systems literature. For each such metric we identify its important properties and explain why is it most appropriate for the given task. For each task we also explain a possible evaluation scenario that can be used to evaluate the various algorithms.

5.1 Predicting Ratings

In this task, the system must provide a set of predicted ratings, and is evaluated on the accuracy of these predictions. This is the most common scenario in the evaluation of regression and classification algorithms in the machine learning and statistics literature (Duda and Hart, 1973; Stone, 1974; Bengio and Grandvalet, 2004). Many evaluation metrics that originated in that literature have been applied here.

Most notably, the Root of the Mean Square Error (RMSE) is a popular method for scoring an algorithm. If $p_{i,j}$ is the predicted rating for user i over item j, and $v_{i,j}$ is the true rating, and $K = \{(i,j)\}$ is the set of hidden user-item ratings then the RMSE is defined as:

$$\sqrt{\frac{\sum_{(i,j)\in K}(p_{i,j}-v_{i,j})^2}{n}}.$$

Other variants of this family are the Mean Square Error (which is equivalent to RMSE) and Mean Average Error (MAE), and Normalized Mean Average Error (NMAE) (Herlocker et al., 2004). RMSE tends to penalize larger errors more severely than the other metrics, while NMAE normalizes MAE by the range of the ratings for ease of comparing errors across domains.

RMSE is suitable for the prediction task, because it measures inaccuracies on all ratings, either negative or positive. However, it is most suitable for situations where we do not differentiate between errors. For example, in the Netflix rating prediction, it may not be as important to properly predict the difference between 1 and 2 stars as between 2 and 3 stars. If the system predicts 2 instead of the true 1 rating, it is unlikely that the user will perceive this as a recommendation. However, a predicted rating of 3 may seem like an encouragement to rent the movie, while a prediction of 2 is typically considered negative. It is arguable that the space of ratings is not truly uniform, and that it can be mapped to a uniform space to avoid such phenomena.

5.2 Recommending Good Items

For the task of recommending items, typically we are only interested in binary ratings, that is, either the item was selected (1) or not (0). Compared to ratings data sets, where users typically rate only a very small number of items, making the data set extremely sparse, binary selection data sets are dense, as each item was either selected or not by the user. An example of such data sets are news story click streams, where we set a value of 1 for each item that was visited, and a value of 0

	Recommended	Not recommended
Preferred	True-Positive (tp)	False-Negative (fn)
Not preferred	False-Positive (fp)	True-Negative (tn)

Table 1: Classification of the possible result of a recommendation of an item to a user.

elsewhere. The task is to provide, given an existing list of items that were viewed, a list of additional items that the user may want to visit.

As we have explained above, these scenarios are typically not symmetric. We are not equally interested in good and bad items; the task of the system is to suggest good items, not to discourage the use of bad items. We can classify the results of such recommendations using Table 1.

We can now count the number of examples that fall into each cell in the table and compute the following quantities:

$$\begin{aligned} \text{Precision} &= \frac{\#\text{tp}}{\#\text{tp}+\#\text{fp}}, \\ \text{Recall (True Positive Rate)} &= \frac{\#\text{tp}}{\#\text{tp}+\#\text{fn}}, \\ \text{False Positive Rate (1 - Specificity)} &= \frac{\#\text{fp}}{\#\text{fp}+\#\text{tn}}. \end{aligned}$$

Typically we can expect a trade off between these quantities—while allowing longer recommendation lists typically improves recall, it is also likely to reduce the precision. In some applications, where the number of recommendations that are presented to the user is not preordained, it is therefore preferable to evaluate algorithms over a range of recommendation list lengths, rather than using a fixed length. Thus, we can compute curves comparing precision to recall, or true positive rate to false positive rate. Curves of the former type are known simply as precision-recall curves, while those of the latter type are known as a Receiver Operating Characteristic[5] or ROC curves.

While both curves measure the proportion of preferred items that are actually recommended, precision-recall curves emphasize the proportion of recommended items that are preferred while ROC curves emphasize the proportion of items that are not preferred that end up being recommended.

We should select whether to use precision-recall or ROC based on the properties of the domain and the goal of the application; suppose, for example, that an online video rental service recommends DVDs to users. The precision measure describes what proportion of their recommendations were actually suitable for the user. Whether the unsuitable recommendations represent a small or large fraction of the unsuitable DVDs that could have been recommended (that is, the false positive rate) may not be as relevant.

On the other hand, consider a recommender system for an online dating site. Precision describes what proportion of the suggested pairings for a user result in matches. The false positive rate describes what proportion of unsuitable candidates are paired with the active user. Since presenting unsuitable candidates can be especially undesirable in this setting, the false positive rate could be the most important factor.

5. A reference to their origins in signal detection theory.

Given two algorithms, we can compute a pair of such curves, one for each algorithm. If one curve completely dominates the other curve, the decision about the winning algorithm is easy. However, when the curves intersect, the decision is less obvious, and will depend on the application in question. Knowledge of the application will dictate which region of the curve the decision will be based on. For example, in the "recommend some good items" task it is likely that we will prefer a system with a high precision, while in the "recommend all good items" task, a higher recall rate is more important than precision.

Measures that summarize the precision recall of ROC curve such as F-measure (Rijsbergen, 1979) and the area under the ROC curve (Bamber, 1975) are useful for comparing algorithms independently of application, but when selecting an algorithm for use in a particular task, it is preferable to make the choice based on a measure that reflects the specific needs at hand.

5.2.1 PRECISION-RECALL AND ROC FOR MULTIPLE USERS

When evaluating precision-recall or ROC curves for multiple test users, a number of strategies that can be employed in aggregating the results. The simplest is to aggregate the hidden ratings from the test set into a set of user-item pairs, generate a ranked list of user-item pairs by combining the recommendation lists for the test users, and then compute the precision-recall or ROC curve on this aggregated data.

This aggregation process assumes that we have a means of comparing recommendations made to different users in order to combine the recommendation lists into a single ranked list. Computing ROC curves in this manner treats the recommendations of different items to each user as being independent detection or classification tasks, and the resulting curve is termed a global ROC (GROC) curve (Schein et al., 2002).

A second approach is to compute the precision and recall (or true positive rate and false positive rate) at each recommendation list length N for each user, and then compute the average precision and recall (or true positive rate and false positive rate) at each N(Sarwar et al., 2000). The resulting curves are particularly valuable because they prescribe a value of N for each achievable precision and recall (or true positive rate and false positive rate), and conversely, can be used to estimate performance at a given N. Thus, this approach is useful in the "recommend some good items" scenario, where one important decision is the length of the recommendation list, by comparing performances along different candidate points along the curves. An ROC curve obtained in this manner is termed a Customer ROC (CROC) curve (Schein et al., 2002).

A third approach is to compute a precision-recall curve (or ROC curve) for each user and then average the resulting curves over users. This is the usual manner in which precision-recall curves are computed in the information retrieval community, and in particular in the influential TREC competitions (Voorhees, 2002b). This method is more relevant in the "recommend all good items" sub-task, if the system provides the user with all available recommendations and the user then scans the list linearly, marking each scanned item as relevant or not. The system can then compute the precision of the items scanned so far, and use the precision recall curve to give the user an estimate of what proportion of the good items have yet to be found.

5.3 Optimize Utility

Estimating the utility of a list of recommendations requires a model of the way users interact with the recommendations. For example, if a movie recommender system presents the DVD cover im-

ages of the top five recommendations prominently arranged horizontally across the top of the screen, the user will probably observe them all and select the items of interest. However, if all the recommendations are presented in a textual list several pages long, the user will probably scan down the list and abandon their scan at some point. In the first case, utility delivered by the top five recommendations actually selected would be a good estimate of expected utility, while in the second case, we would have to model the way users scan lists.

The half-life utility score of Breese et al. (1998) suggested such a model. It postulates that the probability that the user will select a relevant item drops exponentially down the list.

This approach evaluates an unbounded recommendation list, that potentially contains all the items in the catalog. Given such a list we assume that the user looks at items starting from the top. We then assume that an item at position k has a probability of $\frac{1}{2^{(k-1)/(\alpha-1)}}$ of being viewed, where α is a half life parameter, specifying the location of the item in the list with 0.5 probability of being viewed.

In the binary case of the recommendation task the half-life utility score is computed by:

$$\begin{aligned} R_a &= \sum_j \frac{1}{2^{(idx(j)-1)/(\alpha-1)}}, \\ R &= \frac{\sum_a R_a}{\sum_a R_a^{max}}, \end{aligned}$$

where the summation in the first equation is over the preferred items only, $idx(j)$ is the index of item j in the recommendation list, and R_a^{max} is the score of the best possible list of recommendations for user a.

More generally, we can plug any utility function $u(a, j)$ that assigns a value to a user item pair into the half-life utility score, obtaining the following formula:

$$R_a = \sum_j \frac{u(a, j)}{2^{(idx(j)-1)/(\alpha-1)}}.$$

Now, R_a^{max} is the score for the list of the recommendation where all the observed items are ordered by decreasing utility. In applications where the probability that a user will select the idxth item if it is relevant is known, a further generalization would be to use these known probabilities instead of the exponential decay.

5.4 Fixed Recommendations Lists

When users add movies to their queues in Netflix, the system presents a list of 10 movies that they may like. However, when users choose to see recommendations (by clicking "movies that you will love") the system presents all the possible recommendations. If there are too many recommended movies to fit a single page, the system allows the user to move to the next page of recommendations.

These two different usage scenarios illustrate a fundamental difference between recommendation applications—in the first, the system is allowed to show a small, fixed number of recommendations. In the second, the system provides as many recommendations as it can. Even though the two cases match a single task—the "recommend good items" task—there are several important distinctions that arise. It is important to evaluate the two cases properly.

When the system is required to present a list with a small, fixed size, that is known *a priori*, methods that present curves (precision-recall), or methods that evaluate the entire list (half-life

utility score), become less appropriate. For example, a system may get a relatively high half-life utility score, only due to items that fall outside the fixed list, while another system that selects all the items in the list correctly, and uninteresting items elsewhere, might get a lower score. Precision-recall curves are typically used to help us select the proper list length, where the precision and recall reach desirable values.

Another important difference, is that for a small list, the order of items in the list is less important, as we can assume that the user looks at all the items in the list. Moreover, may of these lists are presented in a horizontal direction, which also reduces the importance of properly ordering the items.

In these cases, therefore, a more appropriate way to evaluate the recommendation system should focus on the first N movies only. In the "recommend good items" task this can be done, for example, by measuring the precision at N—the number of items that are interesting out of the recommended N items. In the "optimize utility" task, we can do so by measuring the aggregated utility (e.g., sum of utility) of the items that are indeed interesting within the N recommendations.

A final case is when we have unlimited recommendation lists in the "recommend good items" scenario, and we wish to evaluate the entire list. In this case, one can use the half-life utility score with a binary utility of 1 when the (hidden) item was indeed selected by the user, and 0 otherwise. In that case, the half-life utility score prefers a recommender system that places interesting items closer to the head of the list, but provides an evaluation for the entire list in a single score.

6. Empirical Evaluation

In some cases, two metrics may provide a different ranking of two algorithms. When one metric is more appropriate for the task at hand, using the other metric may result in selecting the wrong algorithm. Therefore, it is important to choose the appropriate evaluation metric for the task at hand.

In this section we provide some empirical examples of the phenomenon we describe above, that is, where different metrics rank algorithms differently. Below, we present examples where algorithms are ranked differently by two metrics, one of which is more appropriate for the task of interest.

6.1 Data Sets

We selected publicly available data sets which were naturally suited to the different recommendation tasks we have described above. We begin by describing the properties of each data set we used.

6.1.1 PREDICTION TASK

For the prediction task we selected two data sets that contained ratings over items—the Netflix data set and the BookCrossing data set. In both cases, the prediction task is quite natural. Users of both systems may want to browse the collection of movies or books, and we would want to offer these users an estimated rating for the presented items.

Netflix: In 2004, the online movie rental company Netflix[6] announced a competition for improving its recommendation system. For the purpose of the competition, Netflix has released a data set containing 480,000 users ratings over 17,700 movies. Ratings are between 1 and 5 stars for each movie. The data set is very sparse—users mostly rated a small fraction of the available movies. In

6. This can be found at `www.netflix.com`.

our experiments, as we are working with simple algorithms, we have reduced the data set to users who rated more than 100 movies, leaving us with 21,179 users, 17,415 movies, and 117 ratings per user on average. Thus, our results are not comparable to results published in the online competition scoreboard.

BookCrossing: The BookCrossing website[7] allows a community of book readers to share their interests in books, and to review and discuss books. Within that system users can provide ratings on the scale of 1 to 10 stars. The specific data set that we used was collected by a 4 week crawl during August and September 2004 (Ziegler et al., 2005). The data set contains 105,283 users and 340,556 books (we used just the subset containing explicit ratings). Average ratings for a user is 10. This data set is even more sparse than the Netflix data set that we used, as there are more items and less ratings per user.

Both data sets share some common properties. First, people watch many movies and read many books, compared with other domains. For example, most people experience with only a handful of laptop computers, and so cannot form an opinion on most laptops. Ratings are also skewed towards positive ratings in both cases, as people are likely to watch movies that they think they will like, and even more so in the case of books, which require a heavier investment of time.

There are also some distinctions between the data sets. Some people feel compelled to share their opinion about books and movies, without asking for a compensation. However, in the Netflix domain, providing ratings makes it easier to navigate the system and rent movies. Therefore, all users of Netflix have an incentive for providing ratings, while only people who like to share their views of books use the BookCrossing system. We can therefore expect that the ratings of the BookCrossing are less representative of the general population of book readers, than the ratings of Netflix user from the general population of DVD renters.

6.1.2 RECOMMENDATION TASK

One instance of the "recommend good items" task is the case where, given a set of items that the user has used (bought, viewed), we wish to recommend a set of items that are likely to be used. Typically, data sets of usage are binary—an item was either used or wasn't used by the user, and the data set is not sparse, because every item is either used or not used by every user. We used here a data set of purchases from supermarket retailer, and a stream of articles that were viewed in a news website.

Belgian retailer: This data set was collected from an anonymous Belgian retail supermarket store, collected over approximately 5 months, in three non-consecutive periods during 1999 and 2000. The data set is divided into baskets, and we cannot detect return users. There are 88,162 baskets, 16,470 distinct items, and 10 items in an average basket. We do not have access for item prices or profits, so we cannot optimize the retailer revenue. Therefore the task is to recommend more items that the user may want to add to the basket.

News click stream: This is a log of click-stream data of an Hungarian online news portal (Bodon, 2003). The data contains 990,002 sessions, 41,270 news stories, and an average of 8 stories for session. The task is, given the news items that a user has read so far, recommend more news items that the user will likely read.

7. This can be found at `www.bookcrossing.com`.

6.1.3 OPTIMIZING UTILITY TASK

Ta-Feng supermarket: A natural example for an application where optimizing utility is important is maximizing the revenues of a retail company. Such companies may provide recommendation for items, hoping that customers following these recommendations will produce higher revenue. In this case, a natural utility function is the revenue (or profit) from the purchase of an item. The Ta-Feng data set (Hsu et al., 2004) contains transaction information collected over 4 months from November, 2000 to February, 2001. There are $32,266$ users and $23,812$ items, where the average number of items bought by a user is 23. In this task, the utility function is the accumulated profit from selling an item—taking into account both the quantity and the profit per item.

6.2 Recommendation Algorithms

As the focus of this survey is on the correct evaluation of recommender systems, and not on sophisticated algorithms for computing recommendation lists, we limit ourselves to a set of very simple collaborative filtering algorithms. We do this because collaborative filtering is by far the most popular recommendation approach, and because we do not believe that it is appropriate to select the evaluation metric based on the recommendation approach (e.g., collaborative filtering vs. content based).

Moreover, we carefully selected algorithms that are better suited for different tasks, so that we could demonstrate that inappropriate choice of evaluation metric can lead to a bad choice of algorithm. As the algorithms that we choose are computationally intensive, we reduced the size of the data set in some cases, in order to reduce the computation time. This should not be done if it was important to realistically simulate the online case. Below, we present the different algorithms and our prior assumptions about their properties.

6.2.1 PEARSON CORRELATION

Typically, the input for a prediction task is a data set consisting of the ratings provided by n users for m items, where $v_{i,j}$ is the rating of user i for item j. Given such a data set, the simplest collaborative filtering method computes the similarity of the active user a to all other users i in the data set, resulting in a score $w(a,i)$. Then, the predicted rating $p_{a,j}$ for a over item j can be computed by:

$$p_{a,j} = \bar{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \bar{v}_i). \tag{1}$$

Perhaps the most popular method for computing the weights $w(a,i)$ is by using the Pearson correlation coefficient (Resnick and Varian, 1997):

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}}$$

where the summations are only over the items that both a and i have rated. To reduce the computational overhead, we use in Equation 1 a neighborhood of size N.

This method is specifically designed for the prediction task, as it computes only a predicted score for each item of interest. However, in many cases people used this method for the recommendation task. This is typically done by predicting the scores for all possible items, and then ordering the items by decreasing predicted scores.

This popular usage may not be appropriate. For example, in the movie domain people may associate ratings with quality, as opposed to enjoyment, which is dependent on external factors such as mood, time of day, and so forth. As such, 5 stars movies may be complicated, requiring a substantial effort from the viewer. Thus, a user may rent many light effortless romantic comedies, which may only get a score of 3 stars, and only a few 5 star movies. While it is difficult to measure this effect without owning a rental store, we computed the average number of ratings for movies with different average rating (Figure 6.2.1). This figure may suggest that movies with higher ratings are not always watched more often than movies with lower ratings. If our assumption is true, a system that recommends items to add to the rental queue by order of decreasing predicted rating, may not do as well as a system that predicts the probability of adding a movie to the queue directly.

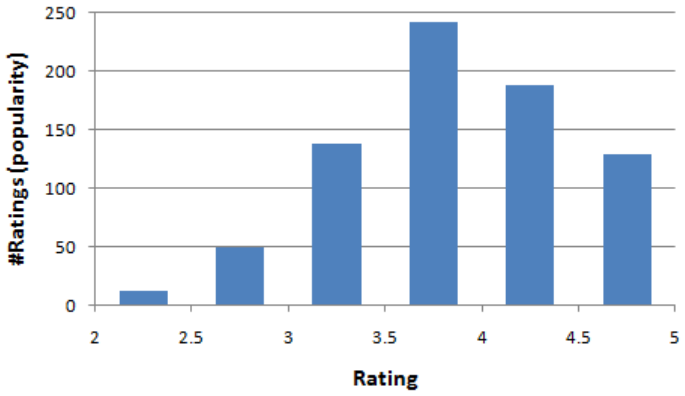


Figure 1: Computing the average number of ratings (popularity) of movies binned given their average ratings.

6.2.2 COSINE SIMILARITY

A second popular collaborative filtering method is the vector similarity metric (Salton, 1971) that measures the cosine angle formed by the two ratings vectors:

$$w(a,i) = \sum_{j \in I_{a,i}} \frac{v_{a,j}}{\sqrt{\sum k \in I_a v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum k \in I_i v_{i,k}^2}}.$$

When computing the cosine similarity, only positive ratings have a role, and negative ratings are discarded. Thus, I_i is the set of items that user i has rated positively and $I_{a,i}$ is the set of items that both users rated positively. Also, the predicted score for a user is computed by:

$$p_{a,j} = \kappa \sum_{i=1}^{n} w(a,i) v_{i,j}.$$

In the case of binary data sets, such as the usage data sets that we selected for the recommendation task, the vector similarity method becomes:

$$w(a,i) = \frac{|I_{a,i}|}{\sqrt{|I_a|} \cdot \sqrt{|I_i|}}$$

where I_a is the set of items that a used, and $I_{a,i}$ is the set of items that both a and i used. The resulting aggregated score can be considered as a non-calibrated measurement of the conditional probability $pr(j|a)$—the probability that user a will choose item j.

In binary usage data sets, the Pearson correlation method would compute similarity using all the items, as each item always has a rating. Therefore, the system would use all the negative "did not use" scores, which typically greatly outnumber the "used" scores. We can therefore expect that Pearson correlation in these cases will result in lower accuracy.

6.2.3 ITEM TO ITEM

The above two methods focused on computing a similarity between users, but another possible collaborative filtering alternative is to focus on the similarity between items. The simplest method for doing so is to use the maximum likelihood estimate for the conditional probabilities of items. Specifically, for the binary usage case, this translates to:

$$pr(j_1|j_2) = \frac{|J_{j_1,j_2}|}{|J_{j_2}|}$$

where J_j is the number of users who used item j, and J_{j_1,j_2} is the number of users that used both j_1 and j_2. While this seems like a very simple estimation, similar estimations are successfully used in deployed commercial applications (Linden et al., 2003).

Typically, an algorithm is given as an input a set of items, and needs to produce a list of recommendations. In that case, we can compute for the conditional probability of each target item given each observed item, and then aggregate the results over the set of given items. In many cases, choosing the maximal estimate has given the best results (Kadie et al., 2002), so we aggregate estimations using a max operator in our experiments.

6.2.4 EXPECTED UTILITY

As optimizing utilities is by far the least explored recommendation task, we choose here to propose a new algorithm that is designed specifically for this task. Intuitively, if the task requires lists that optimize a utility function $u(a, j)$, an obvious method is to order the recommendation by decreasing expected utility:

$$E[j|a] = \tilde{pr}(j|a) \cdot \tilde{u}(a, j)$$

where $\tilde{pr}$ is a conditional probability estimate and $\tilde{u}$ is a utility estimate.

One way to compute the two estimates is by using two different algorithms—a recommendation algorithm for estimating $\tilde{pr}$ and a prediction algorithm for estimating $\tilde{u}$, and then combining their output.

6.3 Experimental Results

Below, we present several examples where different evaluation metrics rank two algorithms differently. We argue that in these cases, using an improper evaluation metric will lead to the selection of an inferior algorithm.

	Netflix	BookCrossing
Pearson	1.07	3.58
Cosine	1.90	4.5

Table 2: RMSE scores for Pearson correlation and Cosine similarity on the Netflix domain (ratings from 1 to 5) and the BookCrossing domain (ratings from 1 to 10).

6.3.1 PREDICTION VS. RECOMMENDATION

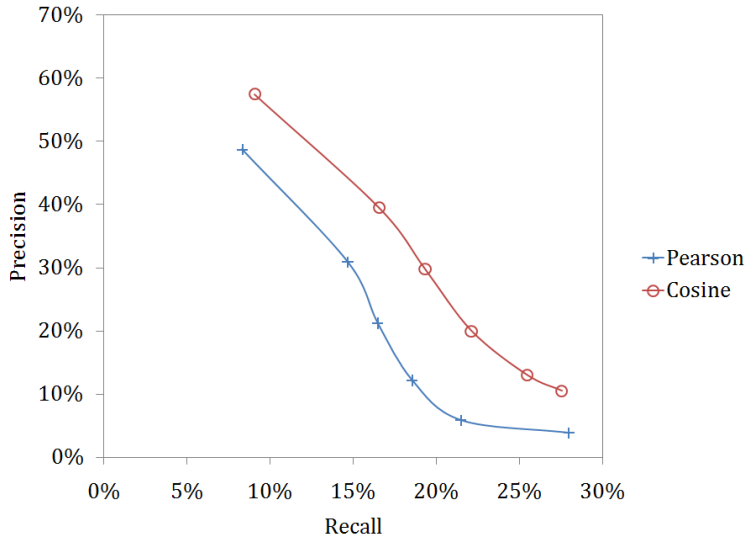
We begin by comparing Pearson correlation and Cosine similarity collaborative filtering algorithms over two tasks—the prediction task and the "recommend good items" task. Both algorithms used neighborhoods consisting of the closest 25 users.

First we evaluated the algorithms in predicting ratings on the Netflix and BookCrossing data sets, where we sampled 2000 test users and a randomly chosen number of test items per test user on each data set. Given Table 2, the algorithm of choice is clear—on both data sets, the predictions given by the Pearson correlation algorithm have lower RMSE scores, and the differences pass a sign test with $p < 0.0001$.

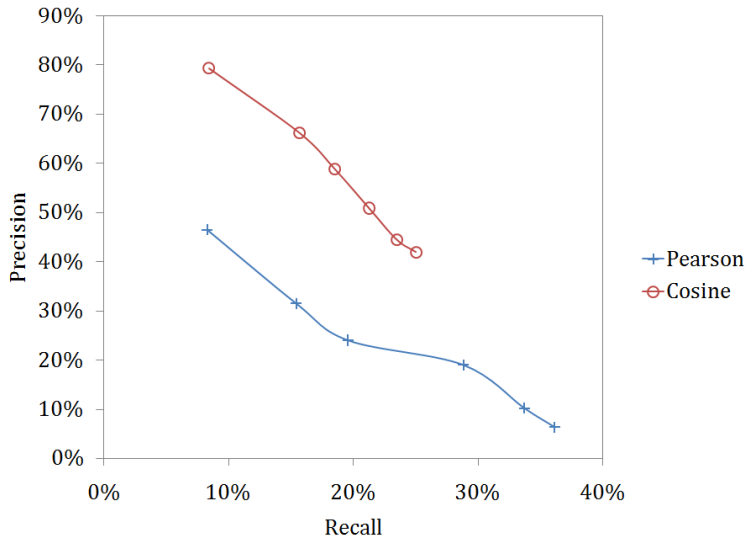
We then evaluated the two algorithms on the recommendation task on the Belgian retailer and news click stream data sets, where we again sampled 2000 test users and a randomly chosen number of test items per test user on each data set. Let us now evaluate the two algorithms on recommendation tasks. To do that, we computed precision-recall curves for the two algorithms on the Belgian retailer data set and the news click stream data set. This was done by computing precision and recall at 1, 3, 5, 10, 25, and 50 recommendations, and averaging the precisions and recalls at each number of recommendations. As Figure 2 shows, in both cases the recommendation lists generated by the Cosine similarity dominate the recommendation lists generated by the Pearson correlation algorithm in terms of precision. In the Belgian retailer data, Cosine similarity also has better recall than Pearson correlation across the board. On the news click stream data, Cosine similarity has better recall than Pearson correlation for 1, 3, and 5, recommendations. All these comparisons were significant according to a sign test with $p < 0.0001$. Therefore, in these cases, one would select the Cosine algorithm as the most appropriate choice.

This experiment shows that an algorithm that is uniformly better at predicting ratings on ratings data sets is not necessarily better at making recommendations on usage data sets. This suggests that it is possible that an algorithm that is better at predicting ratings could be worse at predicting usage in the same domain as well. An interesting experiment would be, given both ratings and usage data over the same users and items, to see whether algorithms that generate recommendation lists by decreasing order of predicted ratings do as well in the recommendation task over the usage data. Unfortunately, we are unaware of any public data set that contains both types of information. Nevertheless, companies such as Amazon or Netflix collect data both of user purchases and of user ratings over items. These companies can therefore make the appropriate decision for the recommendation task at hand.

It is also possible that websites that support multiple recommendation tasks should use different algorithms for the different tasks. For example, the Netflix website contains a prediction task (e.g., for new releases), and two recommendation tasks—a fixed list of recommendations when adding items to the rental queue, and an unlimited list of recommendations in the "movie you will like"



(a) Belgian retailer recommendations



(b) News click stream recommendations

Figure 2: Comparing recommendations generated by Pearson correlation and Cosine similarity. In both cases, the recommendation list is ordered by decreasing predicted score.

section. It may well be that different algorithms that were trained over different data sets (ratings vs. rentals) may rank differently in different tasks. Deciding on the best recommendation engine based solely on RMSE in the prediction task may lead to worse recommendation lists in the two other cases.

6.3.2 RECOMMENDATION VS. UTILITY MAXIMIZATION

In many retail applications, where the retailer is interested in maximizing profits, people may still train their algorithm on usage data solely, and evaluate them using recommendation oriented metrics, such as precision-recall. For example, even though a retailer website wishes to maximize its profit, it may use a binary data set of items that were bought by users to generate recommendations of the type "people who bought this item also bought ...".

To evaluate the performance of such an approach, we used an item-item recommendation system to generate recommendations for the Ta-Feng data set. Alternatively, one can order the items by expected utility. To compute an estimate of expected utility, we interpreted the normalized predicted score of each recommended item as the probability that the user would actually buy that item. In the case where the recommender predicts numerical ratings, we normalize by the highest possible rating and treat the result as a probability distribution. For example, if the highest rating is 5 and the algorithm predicts a score of 4.5 for a specific user we assume the the probability that the user will use the item is 0.9. We then use the average profit earned from each item to predict the profit that would be obtained from each item if the active user bought it. Multiplying the probability that the user would buy an item buy the profit that would result if the user bought the item yielded the required estimate of expected utility.

We then evaluated the two algorithms by comparing their precision-recall curves, which are shown in Figure 3. The curves were generated by evaluating precision and recall at $1, 3, 5, 10, 25$, and 50 recommendations, and averaging the precisions and recalls at each number of recommendations. The averages were computed over 2000 users, and the item-item recommender outperformed the expected profit recommender in terms of both precision and recall at all points with $p < 0.0001$.

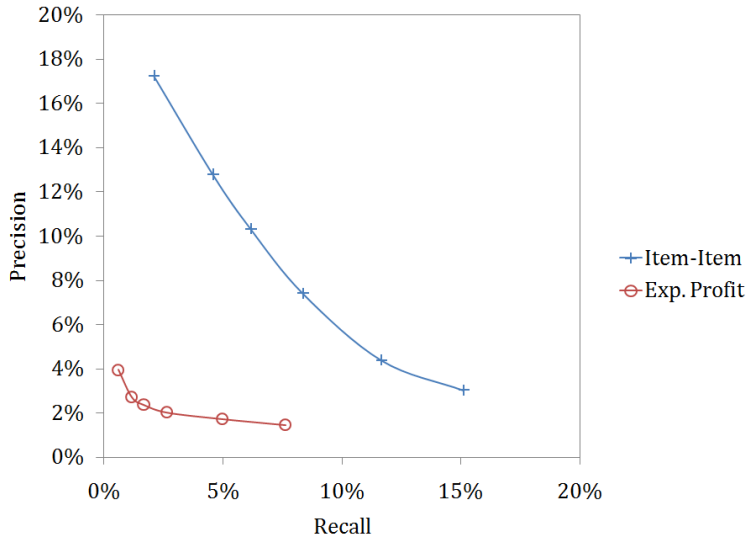


Figure 3: Comparing recommendations generated by the item-item recommender and the expected profit recommender on the Ta-Feng data set.

We then measured an half-life utility score where the utility of a correct recommendation was the profit from selling the correctly recommended item to the user. The results are shown in Table 3.

	Score
Item-Item	0.01
Exp. Profit	0.05

Table 3: Comparing item-item vs. expected utility recommendations on the Ta-Feng data set with the half-life utility score. The utility of a correct recommendation was the profit from selling that item to that user, while the half-life was 5. The trends were similar for other choices of the half-life parameter.

Choosing a recommender based on classification performance would have resulted in an half-life utility score (expected profit) that was 20% of what could have been achieved with the correct choice. This difference is statistically significant with $p < 0.001$.

These results are, of course, not surprising—using a recommender that explicitly attempts to optimize expected profit gives a better expected profit measure. However, occasionally people that are interested in maximizing profit, providing maximum value to the users, or minimizing user effort, use precision-recall to choose a recommendation algorithm.

7. Discussion

Above, we discussed the major considerations that one should make when deciding on the proper evaluation metric for a given task. We now add some discussion, illustrating other conclusions that can be derived, and illuminating some other relevant topics.

7.1 Evaluating Complete Recommender Systems

This survey focuses on the evaluation of recommendation algorithms. However, the success of a recommendation system does not depend solely on the quality of the recommendation algorithm. Such systems typically attempt to modify user behavior which is influenced by many other parameters, most notably, by the user interface. The success of the deployed system in influencing users can be measured through the change in user behavior, such as the number of recommendations that are followed, or the change in revenue.

Decisions about the interface by which users view recommendations are critical to the success of the system. For example, recommendations can be located in different places in the page, can be displayed horizontally or vertically, can be presented through images or text, and so forth. These decisions can make a significant impact, no smaller than the quality of the underlying algorithm, on the success of a system.

When the application is centered around the recommendation system, it is important to select the user interface together with the recommendation algorithm. In other cases, the recommendation system is only a supporting system for the application. For example, an e-commerce website is centered around the item purchases, and a news website is centered around the delivery of news stories. In both cases, a recommender system may be employed to help the users navigate, or to increase sales. It is likely that the recommendations are not the major method for browsing the collection of items.

When the recommender system is only a supporting system, the designer of the application will probably make the decision about the user interface without focusing on positioning recommendations where they have the most influence. In such cases, which we believe to be very common, the developer of the recommender system is constrained by the pre-designed interface, and in many cases can therefore only decide on the best recommendation algorithm, and in some cases perhaps the length of the recommendation list. This paper is targeted at researchers and developers who are making decisions about algorithms, not about the user interface. Designing a good user interface is an interesting and challenging problem, but it is outside the scope of this survey (see, e.g., Pu and Chen 2006).

7.2 Eliciting Utility Functions

A simple way to avoid the need to classify recommendation algorithms, is to assume that we are always optimizing some utility function. This utility function can be the user utility for items (see, e.g., Kumar et al. 1998, Price and Messinger 2005 and Hu and Pu 2009), or it can be the application utility. We can then ask users about their true utility function, or design a utility function that captures the goal of the application, and always choose the algorithm that maximizes this utility.

However, such a view is misleading; eliciting user utilities can be a very difficult task (see, e.g., Braziunas and Boutilier 2005, citealtChajewska and Huang 2008). For example, the value that the user is willing to invest in a laptop depends on a multitude of elements, such as her income, her technical knowledge, the intended use of the laptop and so forth. Indeed, eliciting such functions is the focus of active research, for example by presenting users with forced choices. Therefore, expecting that we will have access to a good estimate of the user utility function is unrealistic— estimating this function may be no easier than coming up with good recommendations.

Furthermore, even when the application designer understands the application utility function, gathering utilities from users may be very difficult. For example, in the Netflix domain, the business model may be to keep the users subscribed. Therefore, the utility of an movie for a user (from the website perspective) is not whether the user enjoys the movie, but rather whether the user will maintain her subscription if the movie is suggested to her. Clearly, most users will not want to answer such questions, and many may not know the answer themselves.

For this reason, when the utility function is unclear, the best we can do is to maximize the number of useful items that we suggest to the user. As such, the recommendation task cannot be viewed as a sub-task for utility optimization.

7.3 Implicit vs. Explicit ratings

Our classification of recommendation tasks sheds some light over another, commonly discussed subject in recommender systems, namely, implicit ratings (Claypool et al., 2001; Oard and Kim, 1998). In many applications, people refer to data that was automatically collected, such as logs of web browsing, or records of product purchases, as an implicit indication for positive opinions over the items that were visited or purchased.

However, this perspective is appropriate only if the task is the prediction task, where we would like to know whether the user will have a positive or negative opinion over certain items. If the task is to recommend more items that the user may buy, given the items that the user has already bought, purchase data becomes an explicit indication. In this case, using ratings that users provide over items is an implicit indication to whether the user will buy the item.

For example, a user may have a positive opinion over many laptop computers, and may rate many laptops highly. However, most people buy only one laptop. In that case, recommending more laptops, based on the co-occurring high ratings, will be inappropriate. However, if we predict the probability of buying a laptop given that another laptop has already been bought, we can expect this probability to be low, and the other laptop will not be recommended.

8. Related Work

In the past, different researchers discussed various topics relevant to the evaluation of recommender systems.

Breese et al. (1998) were probably the first to provide a sound evaluation of a number of recommendation approaches over a collection of different data sets, setting the general framework of evaluating algorithms on more than a single real world data set, and a comparison of several algorithms in identical experiments. The practices that were illustrated in that paper are used in many modern publications.

Herlocker et al. (2004) provide an extensive survey of possible metrics for evaluation. They then compare a set of metrics, concluding that for some pairs of metrics, using both together will give very little additional information compared to using just one.

Another interesting contribution of that paper is a classification of recommendation engines from the user task perspective, namely, what are the reasons and motivations that a user has when interacting with a recommender system. As they are interested in user tasks, and we are interested in the system tasks, our classification is different, yet we share some similar tasks, such as the "recommend some good items" and "recommend all good items" tasks.

Finally, their survey attempted to cover as many evaluation metrics and user task variations as possible, we focus here on the appropriate metrics for the most popular recommendation tasks only.

Mcnee et al. (2003) explain why accuracy metrics alone are insufficient for selecting the correct recommendation algorithm. For example, users may be interested in the serendipity of the recommended items. One way to model serendipity is through a utility function that assigns higher values to "unexpected" suggestions. They also discuss the "usefulness" of recommendations. Many utility functions, such as the inverse log of popularity (Shani et al., 2005) attempt to capture this "usefulness".

Ziegler et al. (2005) focus on another aspect of evaluation—considering the entire list together. This would allow us to consider aspects of a set of recommendations, such as diversification between items in the same list. Our suggested metrics consider only single items, and thus could not be used to evaluate entire lists. It would be interesting to see more evaluation metrics that provide observations over complete lists.

Celma and Herrera (2008) suggest looking at topological properties of the recommendation graph—the graph that connects recommended items. They explain how by looking at the recommendation graph one may understand properties such as the novelty of recommendations. It is still unclear how these properties correlate with the true goal of the recommender system, may it be to optimize revenue or to recommend useful items.

McLaughlin and Herlocker (2004) argue, as we do, that MAE is not appropriate for evaluating recommendation tasks, and that ratings are not necessarily indicative of whether a user is likely to watch a movie. The last claim can be explained by the way we view implicit and explicit ratings.

Some researchers have suggested taking a more holistic approach, and considering the recommendation algorithm within the complete recommendation system. For example, del Olmo and Gaudioso (2008), suggest that systems be evaluated only after deployment, through counting the number of successful recommendations. As we argue above, even in these cases, one is likely to evaluate algorithms offline, to avoid presenting recommendations that have poor quality for users, thus losing their trust.

9. Conclusion

In this paper we discussed how recommendation algorithms should be evaluated in order to select the best algorithm for a specific task from a set of candidates. This is an important step in the research attempt to find better algorithms, as well as in application design where a designer chooses an existing algorithm for their application. As such, many evaluation metrics have been used for algorithm selection in the past.

We review three core tasks of recommendation systems—the prediction task, the recommendation task, and the utility maximization task. Most evaluation metrics are naturally appropriate for one task, but not for the others. We discuss for each task a set of metrics that are most appropriate for selecting the best of the candidate algorithms.

We empirically demonstrate that in some cases two algorithms can be ranked differently by two metrics over the same data set, emphasizing the importance of choosing the appropriate metric for the task, so as not to choose an inferior algorithm.

We also describe the concerns that need to be addressed when designing offline and online experiments. We outline a few important measurements that one must take in addition to the score that the metric provides, as well as other considerations that should be taken into account when designing experiments for recommendation algorithms.

References

D. Bamber. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of Mathematical Psychology*, 12:387–415, 1975.

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

F. Bodon. A fast APRIORI implementation. In *The IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, 2003.

D. Braziunas and C. Boutilier. Local utility elicitation in GAI models. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence*, pages 42–49, Edinburgh, 2005.

J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI: Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

Ò. Celma and P. Herrera. A new approach to evaluating novel recommendations. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, 2008.

M. Claypool, P. Le, M. Waseda, and D. Brown. Implicit interest indicators. In *Intelligent User Interfaces*, pages 33–40. ACM Press, 2001.

F. Hernández del Olmo and E. Gaudioso. Evaluation of recommender systems: A new approach. *Expert Systems Applications*, 35(3), 2008.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7, 2006.

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

C. Goutte and E. Gaussier. A probabilistic interpretation of precision, recall, and F-score, with implication for evaluation. In *ECIR '05: Proceedings of the 27th European Conference on Information Retrieval*, pages 345–359, 2005.

A. Gunawardana and C. Meek. Aggregators and contextual effects in search ad markets. In *WWW Workshop on Targeting and Ranking for Online Advertising*, 2008.

J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1), 2004.

C. N. Hsu, H. H. Chung, and H. S. Huang. Mining skewed and sparse transaction data for personalized shopping recommendation. *Machine Learning*, 57(1-2), 2004.

R. Hu and P. Pu. A comparative user study on rating vs. personality quiz based preference elicitation methods. In *IUI '09: Proceedings of the 13th International Conference on Intelligent User Interfaces*, 2009.

S. L. Huang. Comparision of utility-based recommendation methods. In *The Pacific Asia Conference on Information Systems*, 2008.

C. Kadie, C. Meek, and D. Heckerman. CFW: A collaborative filtering system using posteriors over weights of evidence. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 242–250, San Francisco, CA, 2002. Morgan Kaufmann.

R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1), 2009.

J. A. Konstan, S. M. McNee, C. N. Ziegler, R. Torres, N. Kapoor, and J. Riedl. Lessons on applying automated recommender systems to information-seeking tasks. In *Proceedings of the Twenty-First National Conference on Artifical Intelligence (AAAI)*, 2006.

R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation systems: A probabilistic analysis. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, 1998.

D. Larocque, J. Nevalainen, and H. Oja. A weighted multivariate sign test for cluster-correlated data. *Biometrika*, 94:267–283, 2007.

G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1), 2003.

M. R. McLaughlin and J. L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *SIGIR '04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.

S. Mcnee, S. K. Lam, C. Guetzlaff, J. A. Konstan, and J. Riedl. Confidence displays and training in recommender systems. In *Proceedings of the 9th IFIP TC13 International Conference on Human Computer Interaction INTERACT*, pages 176–183. IOS Press, 2003.

S. M. McNee, J. Riedl, and J. K. Konstan. Making recommendations better: an analytic model for human-recommender interaction. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, 2006.

M. Montaner, B. López, and J. L. De La Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, 19(4), 2003.

D. Oard and J. Kim. Implicit feedback for recommender systems. In *The AAAI Workshop on Recommender Systems*, pages 81–83, 1998.

B. Price and P. Messinger. Optimal recommendation sets: Covering uncertainty over user preferences. In *National Conference on Artificial Intelligence (AAAI)*, pages 541–548. AAAI Press AAAI Press / The MIT Press, 2005.

P. Pu and L. Chen. Trust building with explanation interfaces. In *IUI '06: Proceedings of the 11th International Conference on Intelligent User Interfaces*, 2006.

P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3), 1997.

C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979.

G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

S. L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3), 1997.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, 2000.

J. B. Schafer, J. Konstan, and J. Riedi. Recommender systems in e-commerce. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, 1999.

A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR '02: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

G. Shani, D. Heckerman, and R. I. Brafman. An mdp-based recommender system. *Journal of Machine Learning Research*, 6:1265–1295, 2005.

M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36(1):111–147, 1974.

E. M. Voorhees. The philosophy of information retrieval evaluation. In *CLEF '01: Revised Papers from the Second Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems*, 2002a.

E. M. Voorhees. Overview of trec 2002. In *The 11th Text Retrieval Conference (TREC 2002), NIST Special Publication 500-251*, pages 1–15, 2002b.

C. N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW '05: Proceedings of the 14th International Conference on the World Wide Web*, 2005.